

Container Orchestration and Management II

CNAE – Cloud Native Architecture & Engineering



Prof. Dr.-Ing. Stefan Tai
Sebastian Werner, Elias Grünewald, Nicola Leschke, Maria Borges

Information Systems Engineering
TU Berlin

Energy Efficiency and Sustainability

Can you assess the energy efficiency and sustainability of your application? how?

Task:

Work out different workload scenarios for your application and evaluate at what cost the as-is deployment can respond.

1. What are the costliest workloads and why?

2. Refactor the application to reduce cost for all scenarios while maintain performance.

3. Create a dashboard and/or report by your application.

4. Create a report by your application.

5. Create a report by your application.

6. Create a report by your application.

7. Create a report by your application.

8. Create a report by your application.

9. Create a report by your application.

10. Create a report by your application.

11. Create a report by your application.

12. Create a report by your application.

Architecture:



Tea Store

<https://github.com/DescartesResearch/TeaStore>



Resilience / Fault Tolerance

How can you ensure your system is resilient to faults?

Task:

Determine the resilience of your application and how it can be improved.

1. What are the resilience requirements of your application?

2. Estimate the resilience of your application.

3. Improve the resilience of your application.

4. Create a report by your application.

5. Create a report by your application.

6. Create a report by your application.

7. Create a report by your application.

8. Create a report by your application.

9. Create a report by your application.

Architecture:



SockShop



Performance and Elastic Scalability

How can you ensure your application is scalable and performant?

Task:

Work out different workload scenarios for your application and evaluate at what cost the as-is deployment can respond.

1. What are the costliest workloads and why?

2. Refactor the application to reduce cost for all scenarios while maintain performance.

3. Create a dashboard and/or report by your application.

4. Create a report by your application.

5. Create a report by your application.

6. Create a report by your application.

7. Create a report by your application.

Architecture:



Robot Shop



Privacy and Transparency

How can you enhance the transparency of personal data being processed by your application?

Task:

Find out the transparency requirements of your application and how it can be improved.

1. Which transparency requirements are relevant for your application?

2. Refactor the application to improve transparency.

3. Create a dashboard and/or report by your application.

4. Create a report by your application.

5. Create a report by your application.

Architecture:



Online Boutique

<https://github.com/GoogleCloudPlatform/microservices-demo>



Cost and Performance Management

How can you strike a cost and performance balance in your application?

Task:

Work out different likely workload scenarios for your application and evaluate at what cost the as-is deployment can respond.

1. What are the costliest workloads and why?

2. Refactor the application to reduce cost for all scenarios while maintain performance.

Architecture:



Online Boutique

<https://github.com/GoogleCloudPlatform/microservices-demo>



GameOn!

<https://github.com/gameontext>



Serverless Espresso [AWS]

<https://github.com/aws-samples/serverless-coffee>

Reading Material:

Assignments



1. Review the **06 Assignment slides**,
especially the five refactoring goals and related architectures.
2. Find yourself in a group (see **Group Selection**).
You can also use the forum to find a group.
3. Complete your **formal registration** until 15/05/2023, 23:59 on Moses (MTS).
4. Select the **top 3 projects** (goal+architecture) you want until 12/05/2023 23:59 and
e-mail `n1@ise.tu-berlin.de`. Include the group number, CC all group members and name three
goals and corresponding architectures you like/dislike.
5. On **16/05/2023**, we will get back to you with the **topic assignment**.
6. On **02/06/2023**, each group will have the chance for **individual consultation**.
7. On **12/06/2023**, you must upload the **Design Document** and **presentation slides**.



06_Assignment Hochgeladen 9.05.2023 11:34



Group Selection

Geöffnet: Dienstag, 9. Mai 2023, 11:00

Schließt: Freitag, 12. Mai 2023, 23:59



 Nicht verfügbar, es sei denn: Sie sind in **Zugelassen**

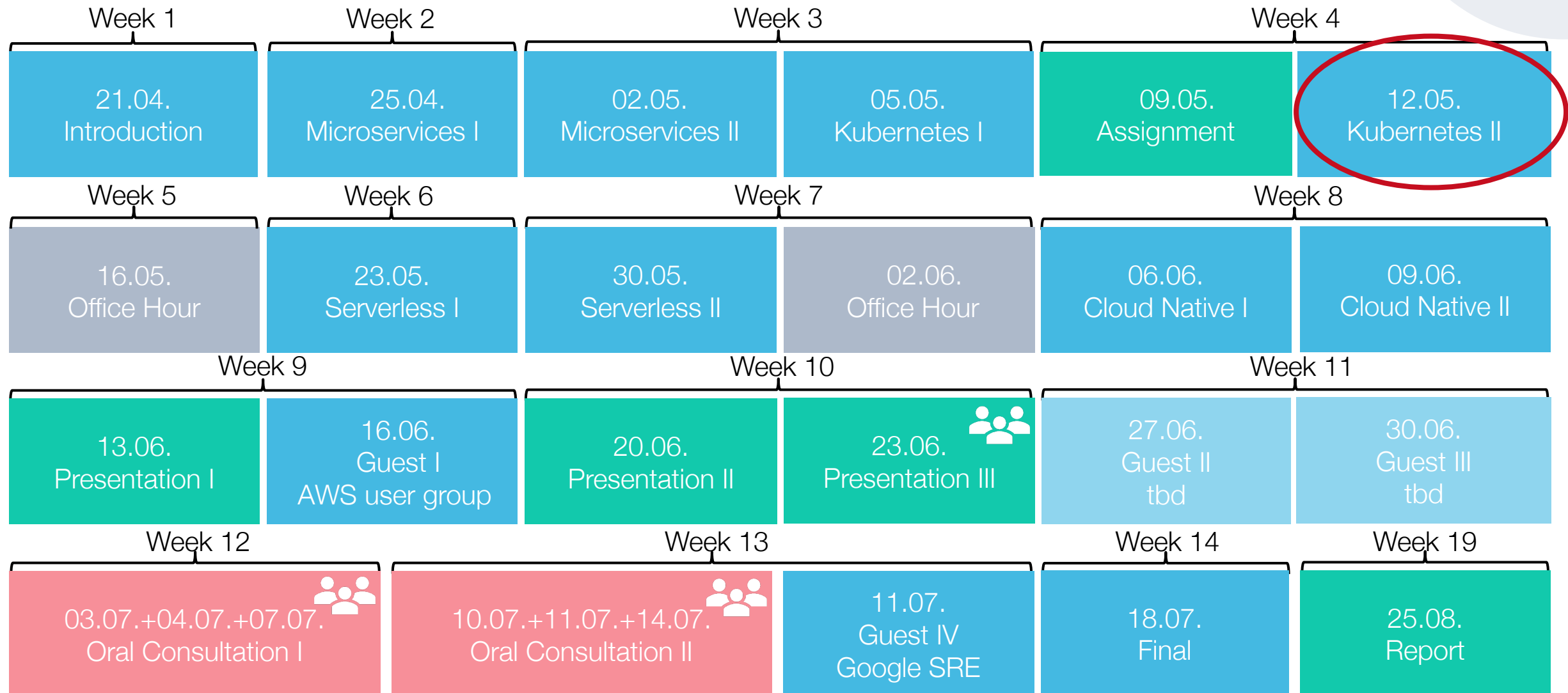
Formal registration deadline: **15.05.2023, 23:59**

Registration via Moses:

<https://moseskonto.tu-berlin.de/moses/modultransfersystem/index.html>

- select "MTS / Module exams registration/deregistration" after logging in
- Select right course and semester!
- instructions with screenshots are only available in [German](#)

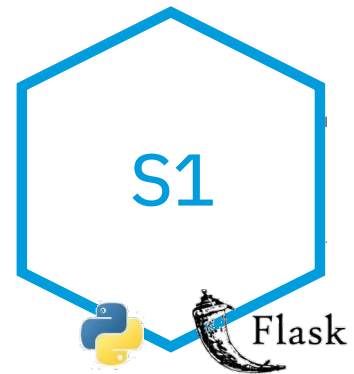
Schedule (still preliminary)



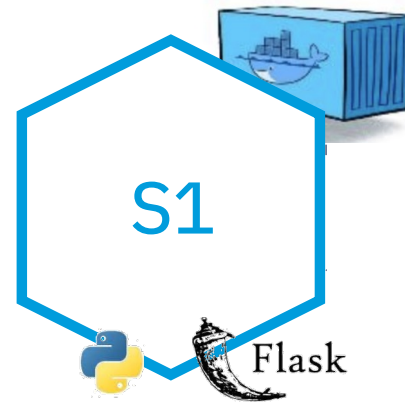
Recap: Container orchestration and management I

- Containerization allows for applications to be **deployed consistently across different environments**.
- Container orchestration platform provide a **centralized interface** for managing containerized applications (incl. deployment, updates, ...).
- Container orchestration platforms can perform **resource optimizations** (incl. scaling) and ensure availability (incl. self-healing mechanisms).
- Container orchestration platforms are still **complex to manage**, esp. in large-scale environments with many non-functional requirements.

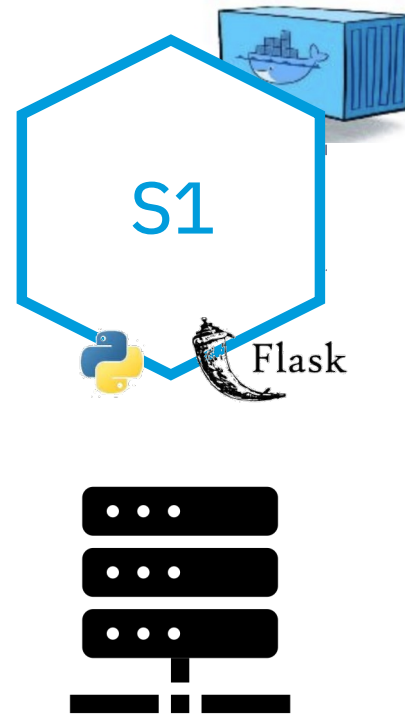
Service development



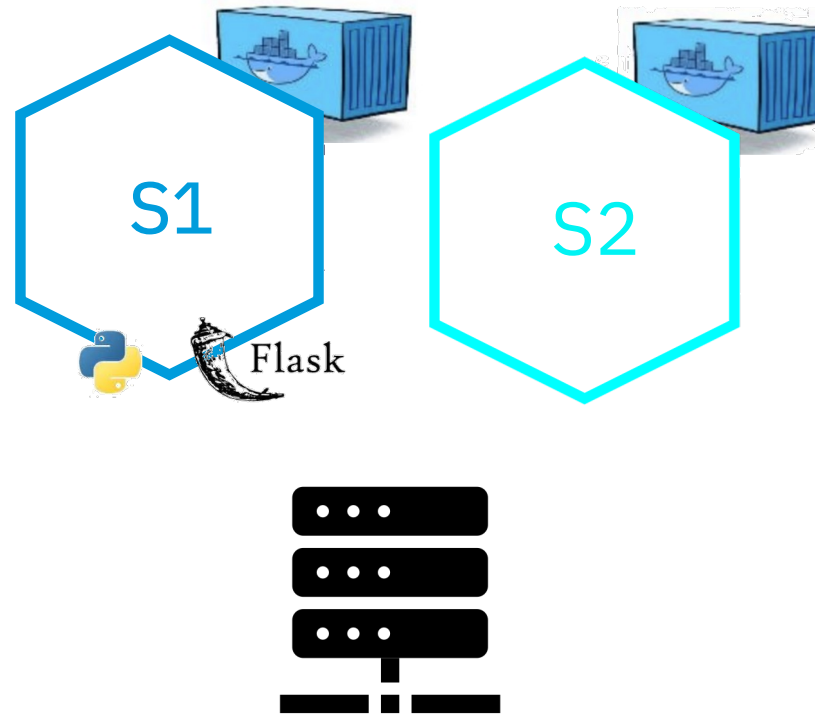
Containerization



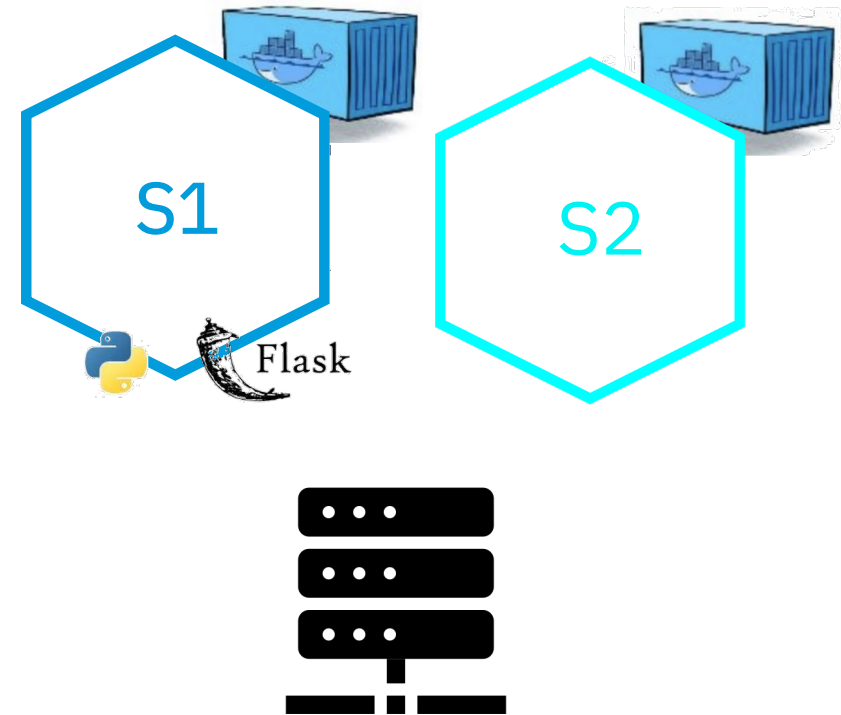
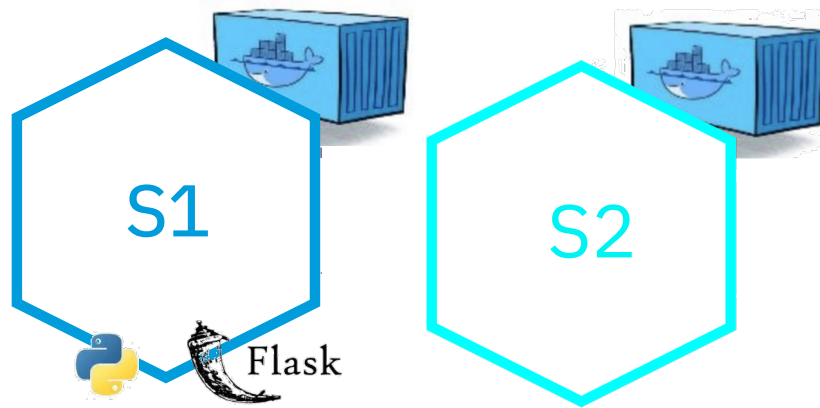
Deployment, e.g., on a VM



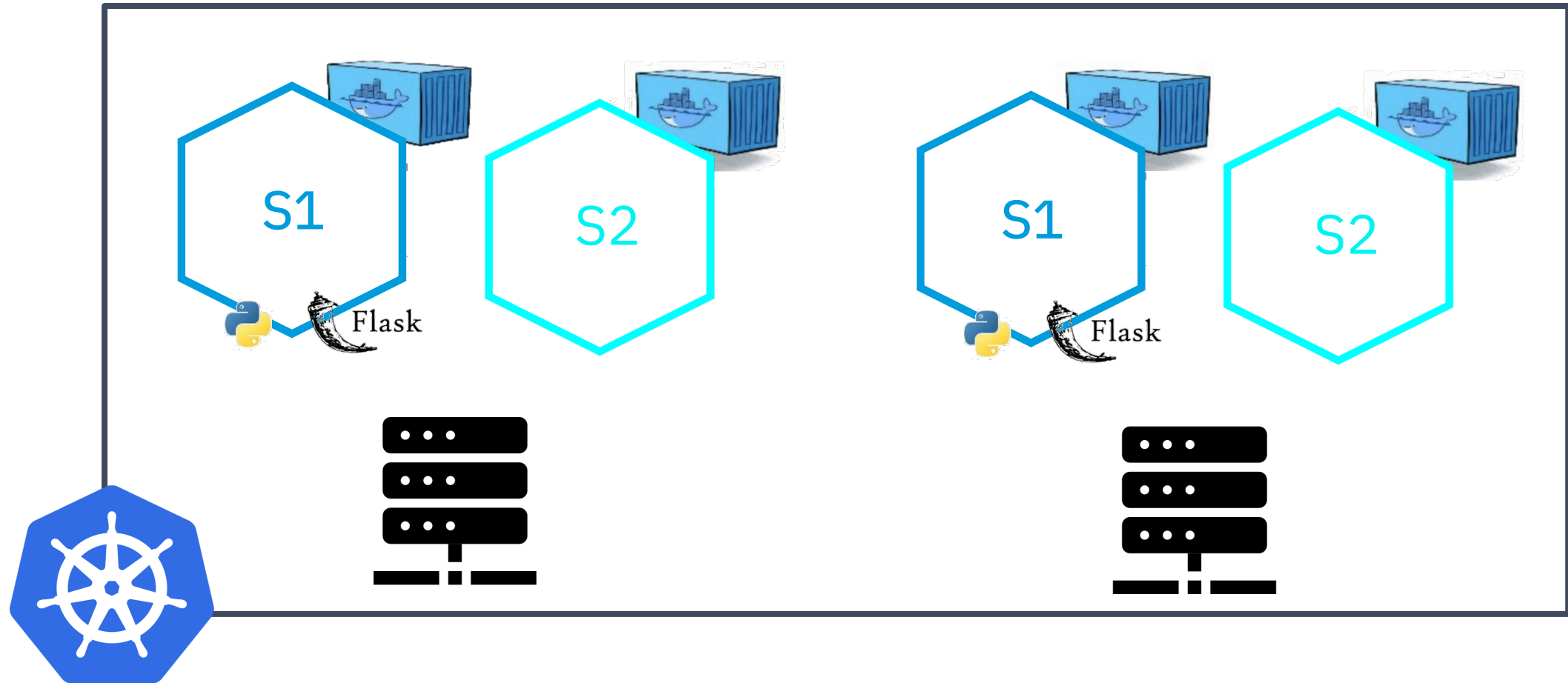
Multi-service environments



Horizontal scaling



Container orchestration



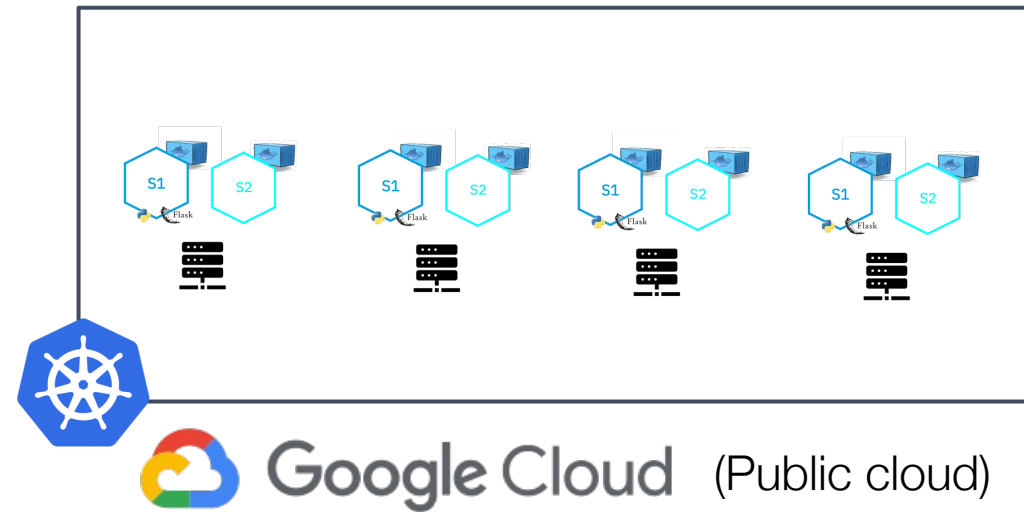
Auto-scaling



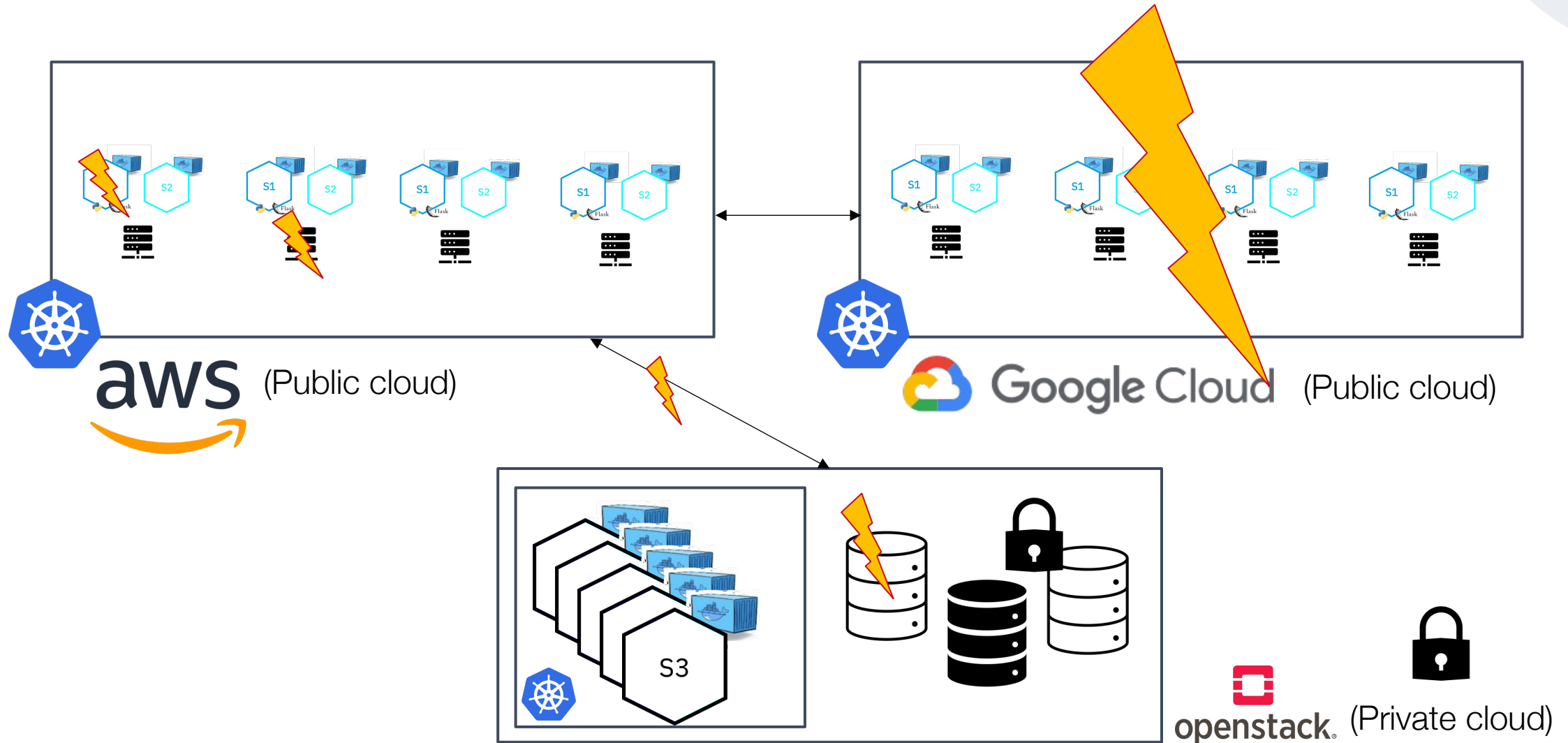
Multi-cloud environments



Hybrid & Multi-cloud environments



Errors and failures are the norm



Agenda



Container orchestration and management I

Containerization, Container orchestration, Case studies, Challenges

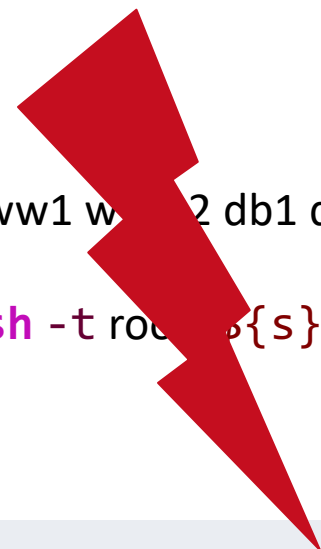
Container orchestration and management II

Infrastructure as Code, Release strategies, Service Discovery

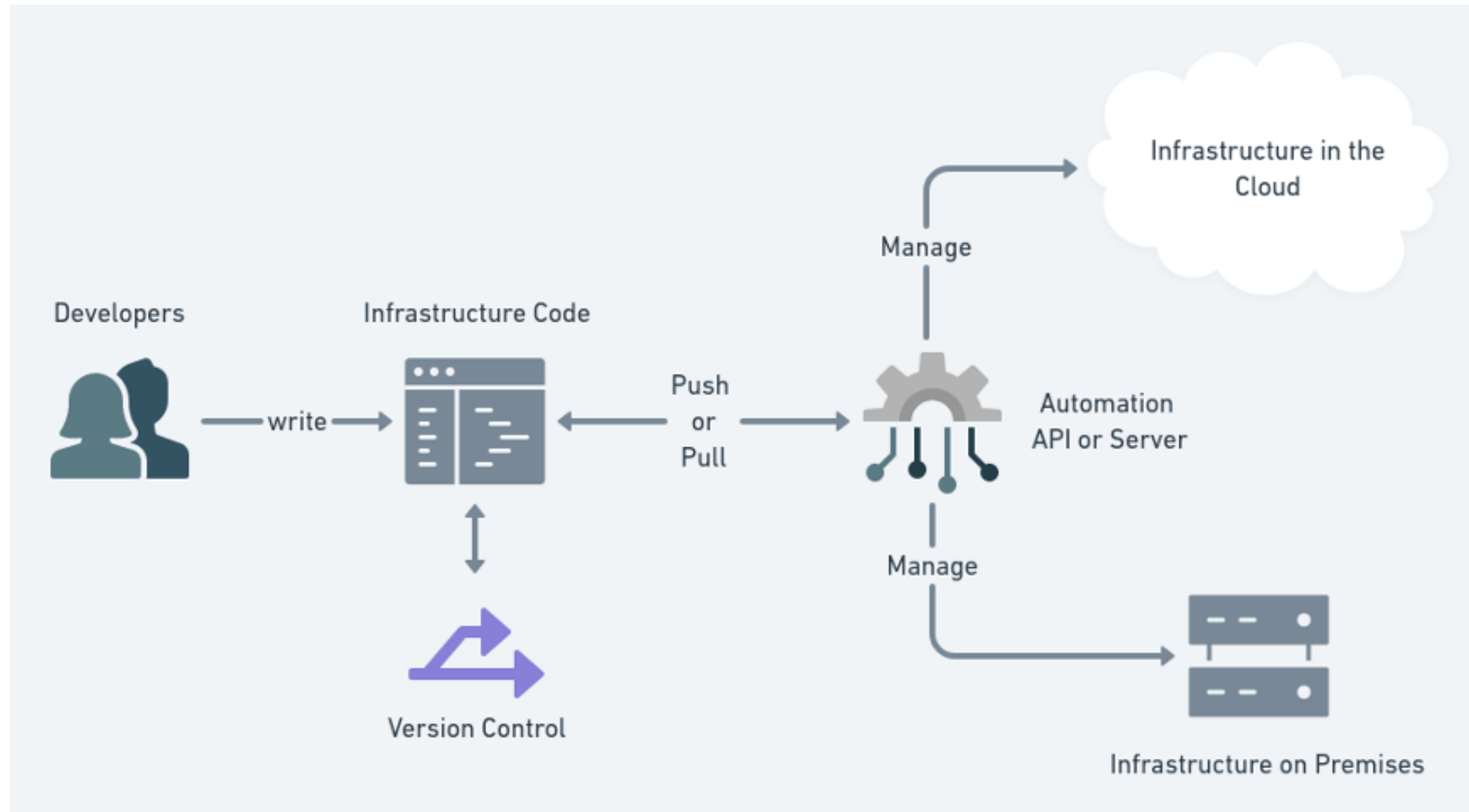
Errors and failures are the norm



```
for s in www1 www2 db1 db2 cache1:
do
ssh -t root@{s} "shutdown -r now"
done
```



Infrastructure as Code



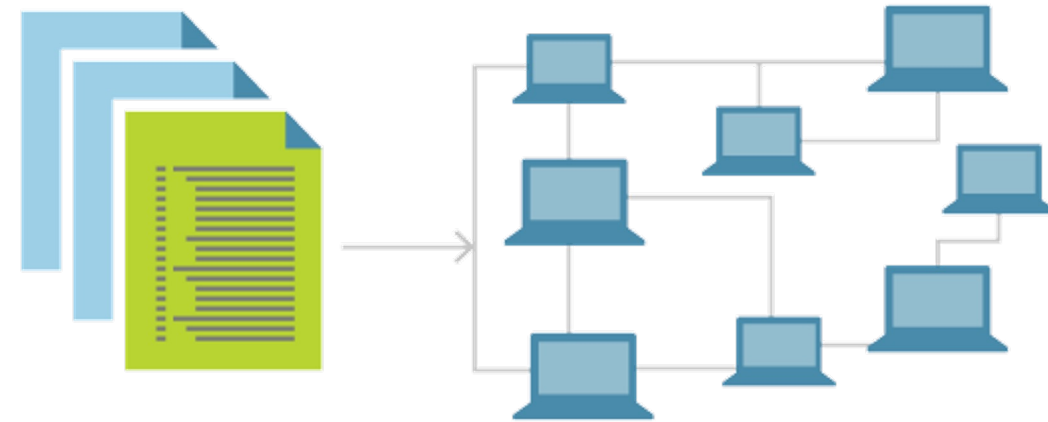
<https://blog.ceteris.ag/infrastructure-as-code/>

Infrastructure as Code (IaC)

- By using a high-level programming language or **declarative configuration** files, infrastructure resources can be defined, provisioned, and managed efficiently.
- **Automation of infrastructure** changes allows tracking and auditing, simplifying the management of complex environments.
- Organizations can benefit from **faster deployment times** and **improved consistency** (e.g., wrt. the resources used).
- IaC can promote collaboration between development and operations teams by providing a **common language and tooling** for managing resources.

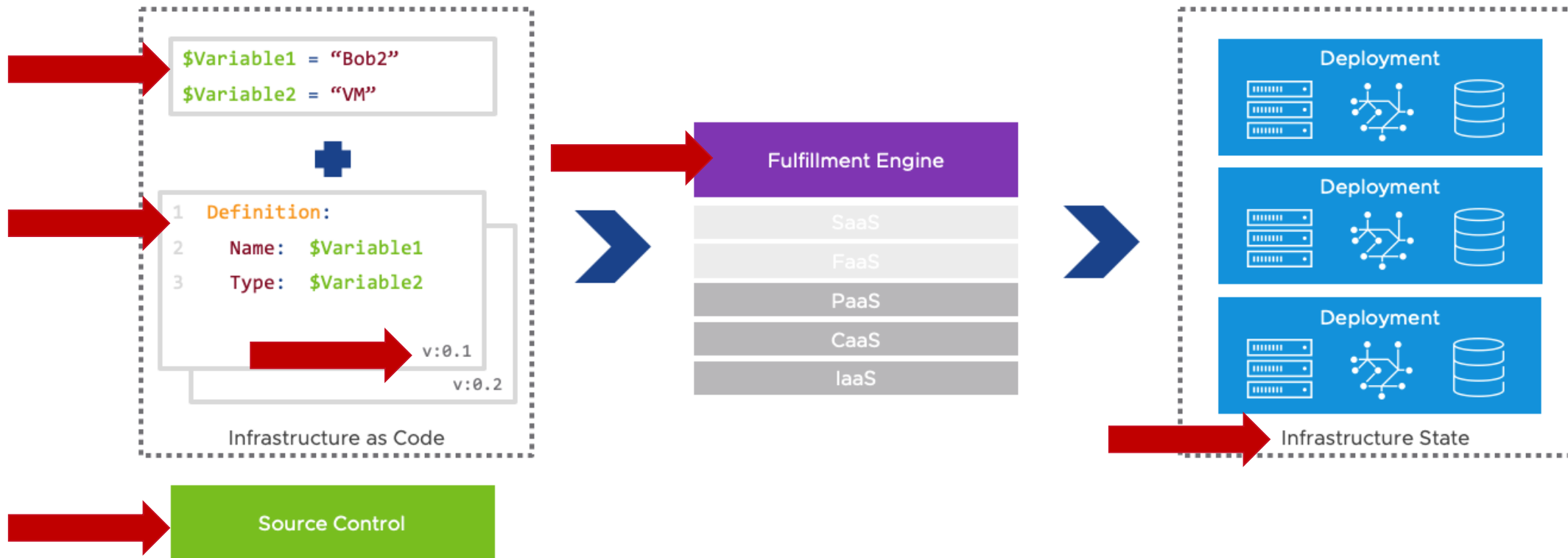
Declarative infrastructure

- Declarative infrastructure definitions describe the **desired state of infrastructure resources**.
- When applied, IaC tools (such as Ansible, Chef, Puppet, or Terraform) evaluate the current state of infrastructure and determine **what changes are necessary** to bring it into alignment with the desired state.
- Declarative configuration files are **easier to read and understand** than traditional imperative approaches.
- They are (more) **idempotent**, resulting in the same desired state when run multiple times, reducing the risk of unintended changes or inconsistencies.



<https://blog.coffeeapplied.com/7-principles-of-infrastructure-as-code-on-azure-and-beyond-51842e13b00>

Infrastructure code



<https://www.ayedo.de/posts/wie-infrastructure-as-code-devops-verbessert/>

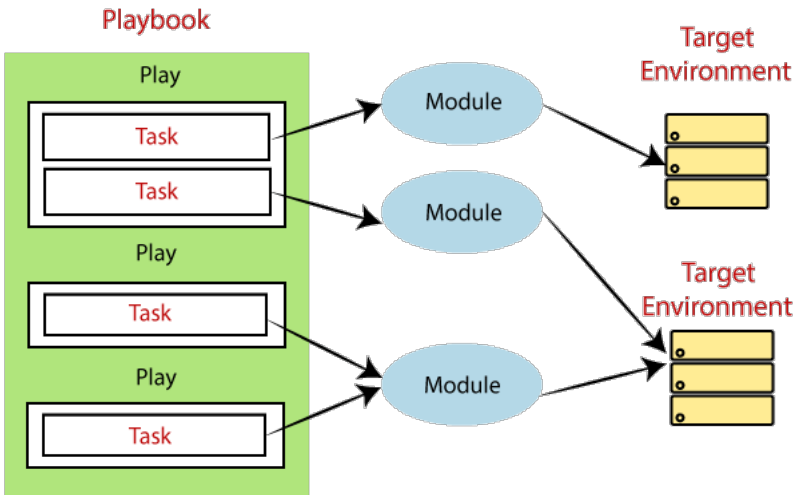
Ansible example

A file (inventory.yaml) which lists the hostnames/IP addresses/... of your (virtual) machines, may be grouped ("virtualmachines", "bare metal", "machines in Berlin")

- **name:** My first play
- hosts:** virtualmachines
- tasks:**

- **name:** Ping my hosts
- ansible.builtin.ping:**
- **name:** Print message
- ansible.builtin.debug:**
- msg:** Hello world

modules



<https://www.javatpoint.com/ansible-playbooks>

```
$ ansible-playbook -i inventory.yaml playbook.yaml
```

Playbook: A list of plays that define the order in which Ansible performs operations, from top to bottom.

Play: An ordered list of tasks that maps to managed nodes in an inventory.

Task: A list of one or more modules that defines the operations that Ansible performs.

Module: A unit of code or binary that Ansible runs on managed nodes.

https://docs.ansible.com/ansible/latest/getting_started/get_started_playbook.html

Ansible modules



The screenshot shows the Ansible documentation website for the `aws_eks_cluster` module. The page title is "aws_eks_cluster – Manage Elastic Kubernetes Service Clusters". It includes a sidebar with navigation links, a main content area with sections for Synopsis, Requirements, and Parameters, and a table for the Parameters section.

aws_eks_cluster – Manage Elastic Kubernetes Service Clusters

New in version 2.7.

- Synopsis
- Requirements
- Parameters
- Notes
- Examples
- Return Values
- Status

Synopsis

- Manage Elastic Kubernetes Service Clusters

Requirements

The below requirements are needed on the host that executes this module.

- boto
- boto3
- botocore
- python >= 2.6

Parameters

Parameter	Choices/Defaults	Comment
aws_access_key		AWS access key. If not set then the value

- **name:** Create an EKS cluster

aws_eks_cluster:

name: my_cluster

version: v1.10.0

role_arn: my_eks_role

subnets:

- subnet-aaaa1111

security_groups:

- my_eks_sg

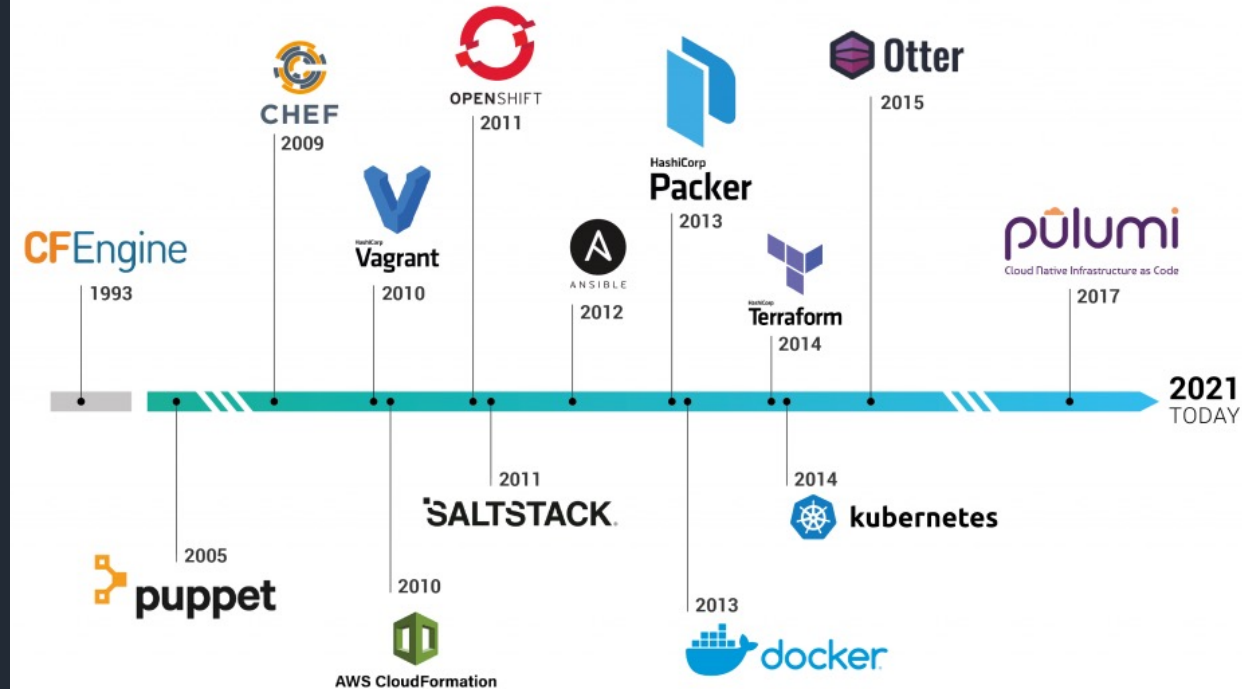
- sg-abcd1234

register: caller_facts ← will be available as a variable in your playbooks

Tooling

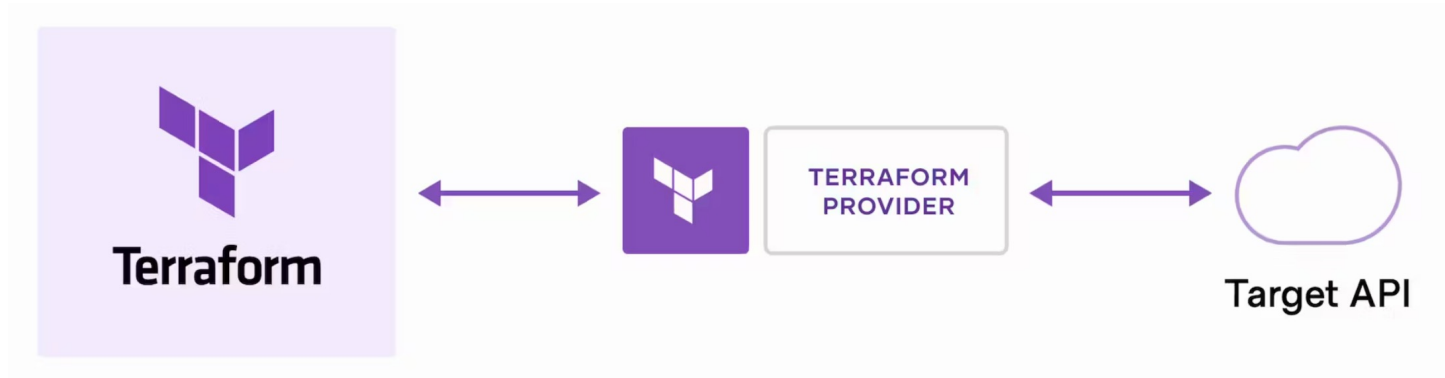
IaC since the early 2000s and they are here to stay.

Cloud platforms accelerated the adoption, esp. with growing infrastructures.

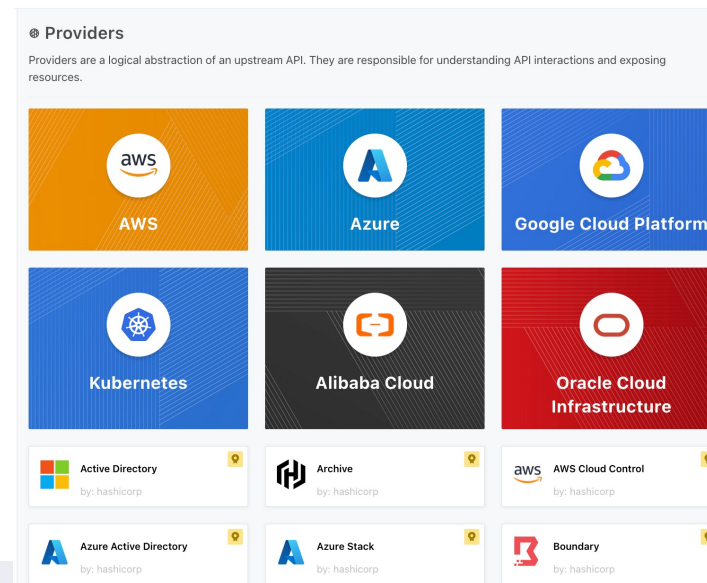


<https://www.novatec-gmbh.de/en/blog/understanding-infrastructure-as-code-iac-in-less-than-10-minutes/>

Terraform

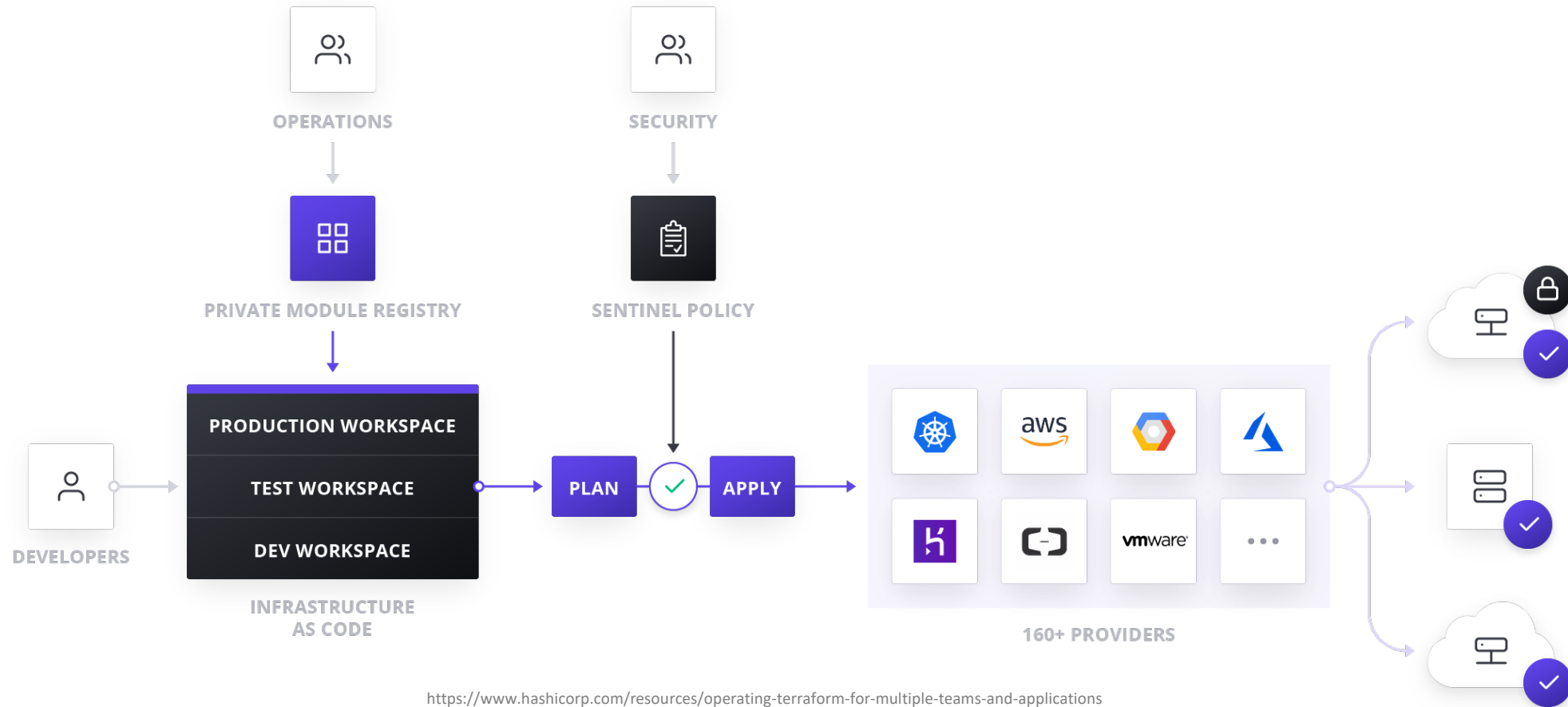


<https://developer.hashicorp.com/terraform/intro>



<https://registry.terraform.io/browse/providers>

Terraform workflow



Terraform workflow (contd.)



```
# Create GKE cluster
resource "google_container_cluster" "my_cluster" {

  name      = var.name
  location  = var.region

  node_pool {
    initial_node_count = var.node_count
    node_config {
      image_type = "COS_CONTAINERD"
      machine_type = "n1-standard-8"
      service_account = google_service_account.default.email
      oauth_scopes = [
        "https://www.googleapis.com/auth/cloud-platform"
      ]
      metadata = {
        disable-legacy-endpoints = "true"
      }
    }
  }
}
```

Write

```
# null_resource.apply_deployment will be created
+ resource "null_resource" "apply_deployment" {
  + id = (known after apply)
}

# null_resource.wait_conditions will be created
+ resource "null_resource" "wait_conditions" {
  + id = (known after apply)
}

# module.gcloud.null_resource.additional_components[0] will be created
+ resource "null_resource" "additional_components" {
  + id = (known after apply)
  + triggers = {
    + "additional_components_command" = ".terraform/modules/gcloud/sc
    + "arguments" = "810a7c5e7880f8aabbf8d16ab8239
    + "md5" = "7f22bf39aaf5ce5980111c3587be
  }
}

# module.gcloud.null_resource.additional_components_destroy[0] will be cr
+ resource "null_resource" "additional_components_destroy" {
  + id = (known after apply)
  + triggers = {
    + "additional_components_command" = ".terraform/modules/gcloud/sc
  }
}

# module.gcloud.null_resource.run_command[0] will be created
+ resource "null_resource" "run_command" {
  + id = (known after apply)
  + triggers = {
    + "arguments" = "810a7c5e7880f8aabbf8d16ab8239bbcc"
    + "create_cmd_body" = "container clusters get-credentials o
    + "create_cmd_entrypoint" = "gcloud"
    + "gcloud_bin_abs_path" = "/google-cloud-sdk/bin"
    + "md5" = "7f22bf39aaf5ce5980111c3587bef5b5"
  }
}
```

Plan

```
version": 4,
terraform_version": "1.3.9",
serial": 4,
lineage": "61d4d52f-ae1a-bb4f-abd8-5aa724b27d49",
outputs": {
  "cluster_location": {
    "value": "us-central1",
    "type": "string"
  },
  "cluster_name": {
    "value": "online-boutique",
    "type": "string"
  }
},
resources": [
  {
    "mode": "managed",
    "type": "google_container_cluster",
    "name": "my_cluster",
    "provider": "provider[\"registry.terraform.io/hashicorp/google\"]",
    "instances": [
      {
        "schema_version": 1,
        "attributes": {
          "addons_config": null,
          "authenticator_groups_config": null,
          "binary_authorization": [],
          "cluster_autoscaling": null,
          "cluster_ipv4_cidr": null,
          "confidential_nodes": null,
```

Apply

Advanced example



```
47 module "eks" {
48   source      = "terraform-aws-modules/eks/aws"
49   cluster_name = local.cluster_name
50   cluster_version = "1.21"
51   subnets    = module.vpc.private_subnets
52   vpc_id      = module.vpc.vpc_id
53   version     = "15.1.0"
54
55   workers_group_defaults = {
56     root_volume_type = "gp2"
57   }
58
59   worker_groups = [
60     {
61       name           = "${local.cluster_name}_worker_group"
62       instance_type  = "t2.small"
63       asg_desired_capacity = 3
64     }
65   ]
66 }
```

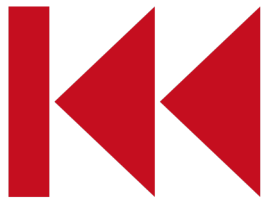
```
19 resource "azurerm_kubernetes_cluster" "default" {
20   name                = "${random_pet.prefix.id}-aks"
21   location            = azurerm_resource_group.default.location
22   resource_group_name = azurerm_resource_group.default.name
23   dns_prefix          = "${random_pet.prefix.id}-k8s"
24
25   default_node_pool {
26     name        = "default"
27     node_count  = 3
28     vm_size     = "Standard_D2_v2"
29     os_disk_size_gb = 30
30   }
31
32   service_principal {
33     client_id = var.appId
34     client_secret = var.password
35   }
36
37   role_based_access_control_enabled = true
38 }
```



Amazon EKS



Azure

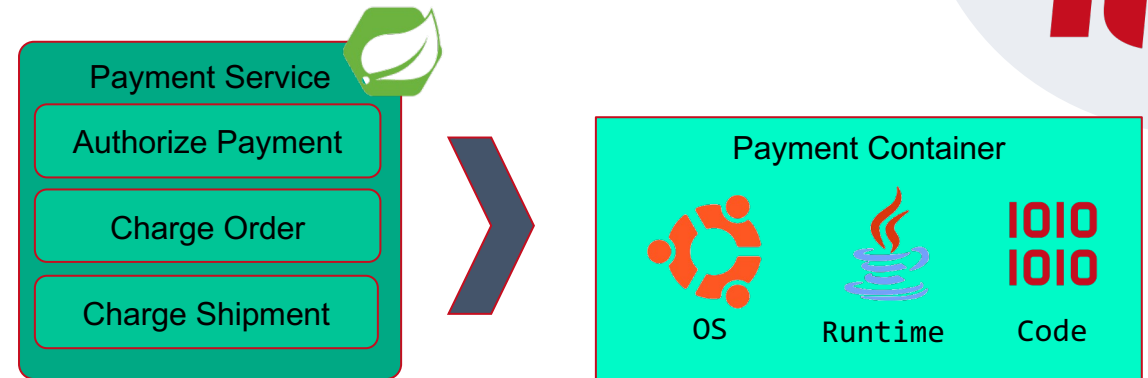


Deployment

- Each **Service** is packaged as a container
- We need to configure each Service as a **Deployment** of containers
- Databases and infrastructure service can:
 - run inside Kubernetes,
 - or be located externally

More than one container
can be part of a service

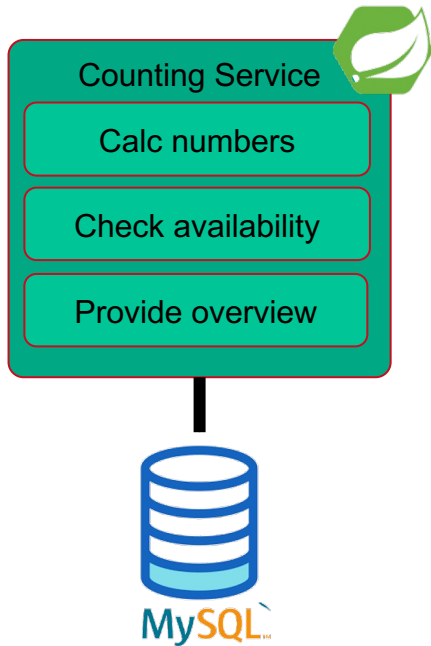
Deployment Package:



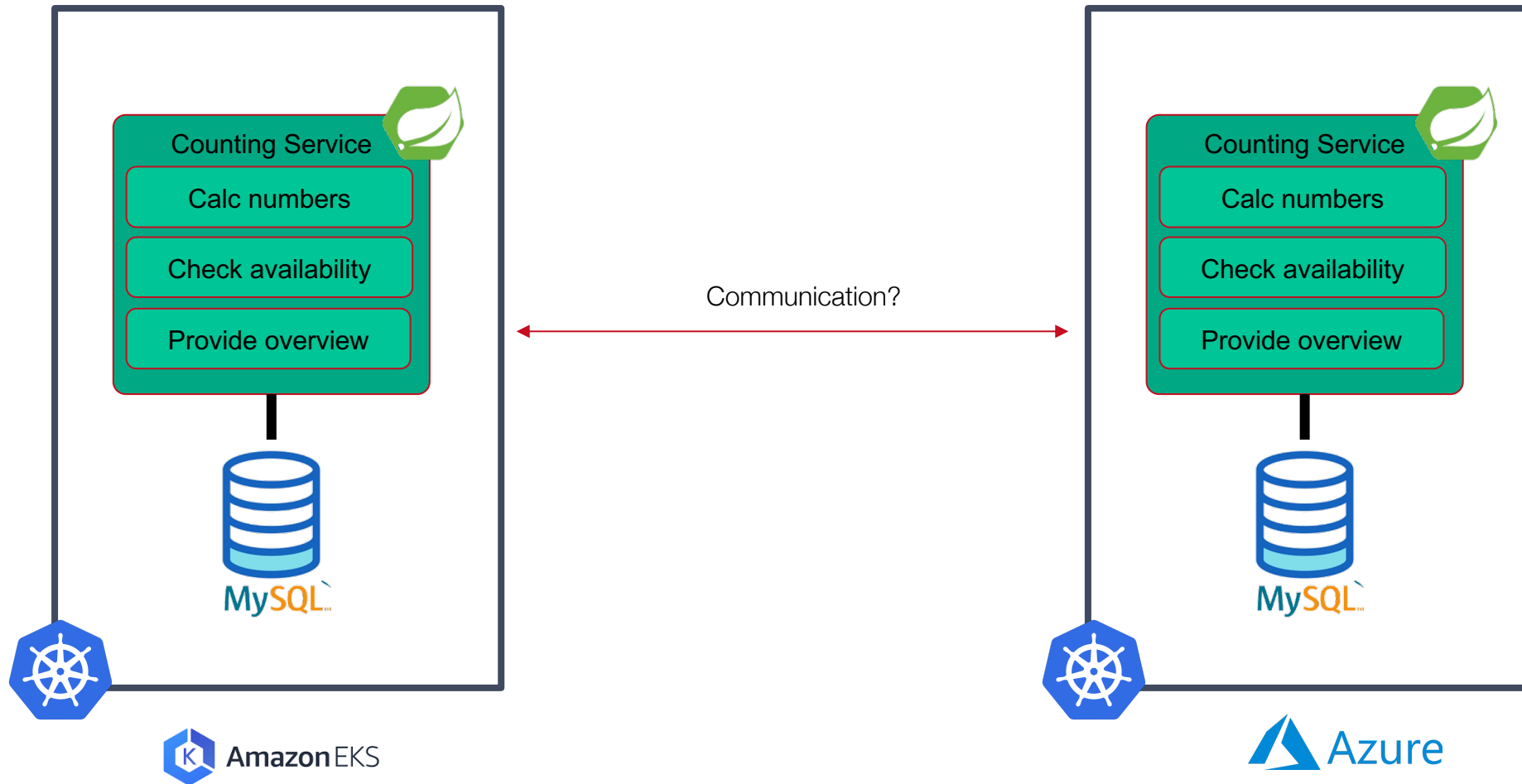
Deployment Configuration:

```
kind: Service
metadata:
  name: payment
spec:
  spec:
    ports:
      - port: 80
        protocol: TCP
    selector:
      run: payment-dep
  ----
kind: Deployment
metadata:
  name: payment-dep
spec:
  replicas: 2
  template:
    spec:
      containers:
        - name: payment-od
          image: payment:v1
```

payment.yaml



```
52 resource "kubernetes_service" "counting" {
53   provider = kubernetes.aks
54   metadata {
55     name      = "counting"
56     namespace = "default"
57     labels = {
58       "app" = "counting"
59     }
60   }
61   spec {
62     selector = {
63       "app" = "counting"
64     }
65     port {
66       name      = "http"
67       port      = 9001
68       target_port = 9001
69       protocol  = "TCP"
70     }
71     type = "ClusterIP"
72   }
73 }
```

Cluster federation with Consul and Terraform

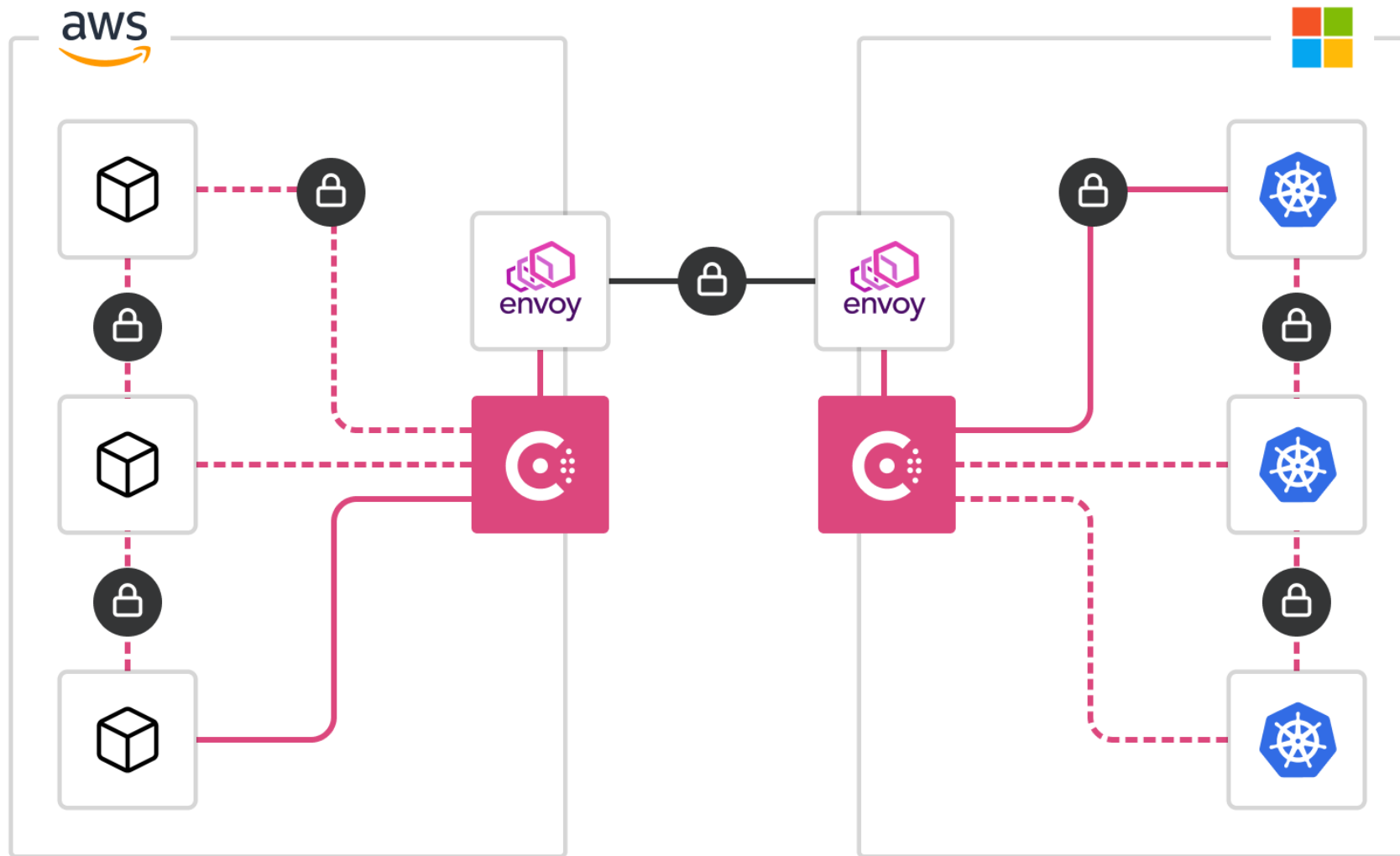


```

49  resource "helm_release" "consul_dc1" {
50      provider    = helm.eks
51      name        = "consul"
52      repository  = "https://helm.releases.hashicorp.com"
53      chart       = "consul"
54      version     = "1.0.2"
55
56      values = [
57          file("dc1.yaml")
58      ]
59  }

```

Advanced example



<https://www.hashicorp.com/products/consul/multi-platform-service-mesh>

Typical Terraform files

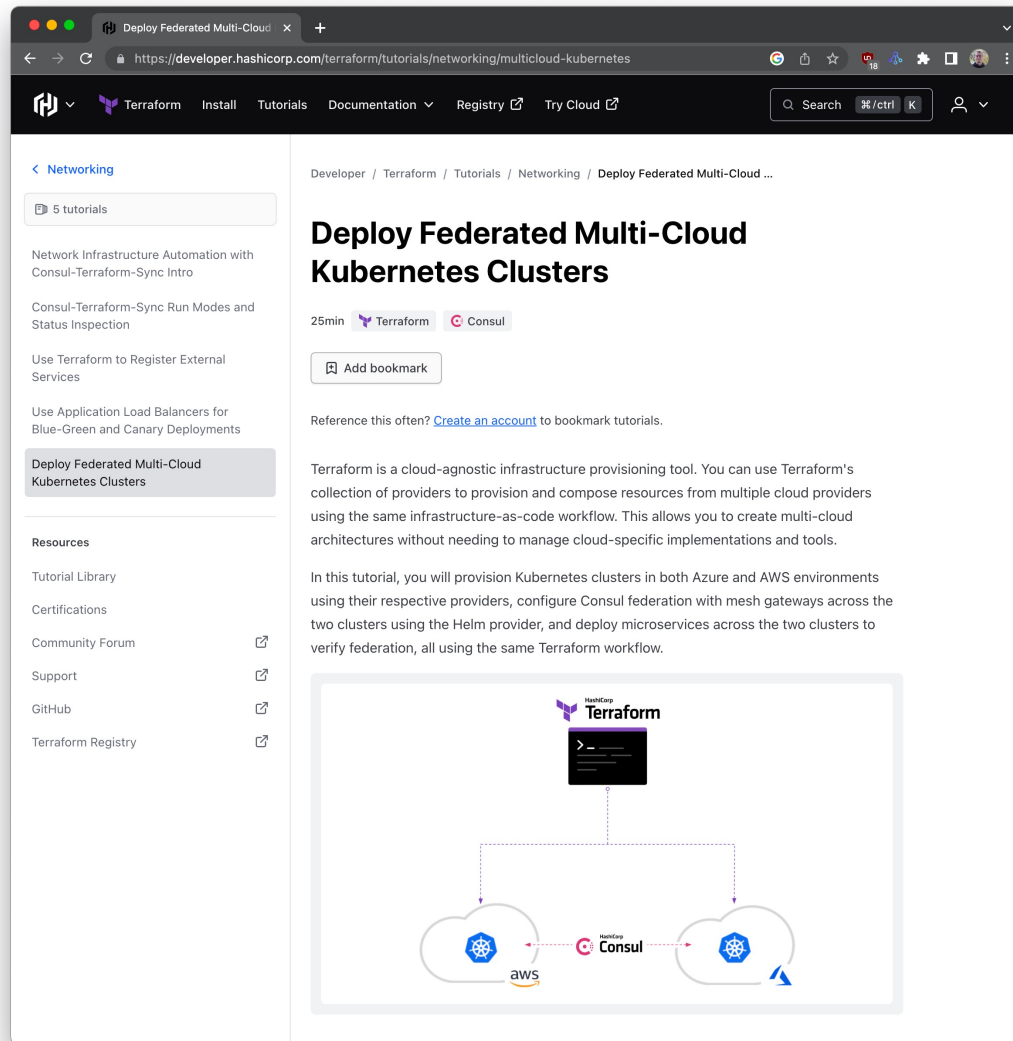
`main.tf` → includes most definitions (perhaps all)

`variables.tf` → definition of global variables

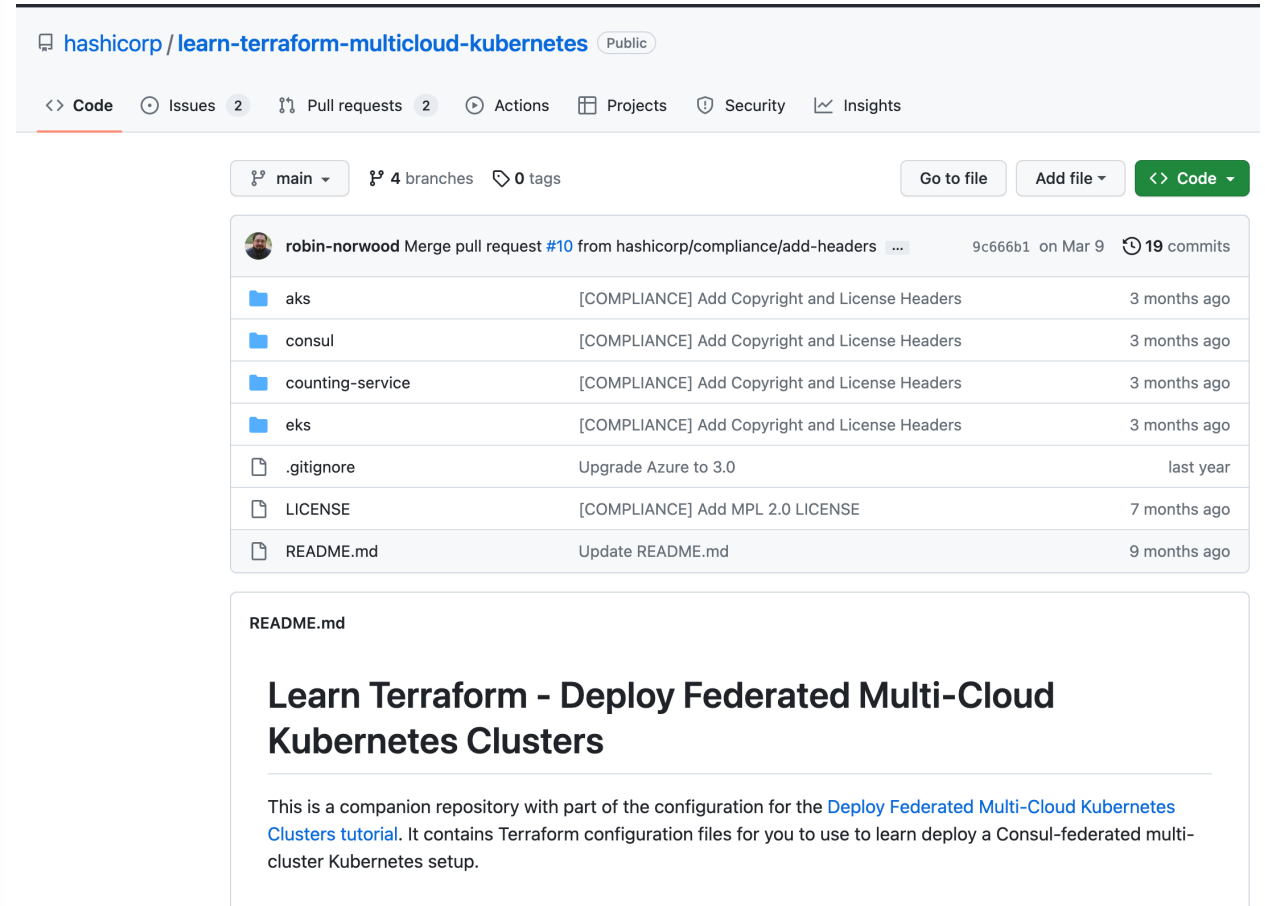
`output.tf` → defines which information should be printed/stored after `apply`

`providers.tf` → define provider configurations

Advanced example



The screenshot shows the HashiCorp Developer website. The main content area is titled "Deploy Federated Multi-Cloud Kubernetes Clusters" and includes a brief introduction to Terraform and the tutorial's goal. A diagram illustrates the architecture: Terraform is used to provision resources in both AWS and Azure cloud environments. These environments are connected via Consul for federation, and Kubernetes clusters are deployed in each cloud.



The screenshot shows the GitHub repository page for "hashicorp/learn-terraform-multicloud-kubernetes". It displays the repository's structure, including folders for different Kubernetes providers (aks, consul, counting-service, eks) and files for .gitignore, LICENSE, and README.md. The README.md file is expanded, showing the title "Learn Terraform - Deploy Federated Multi-Cloud Kubernetes Clusters" and a description of the repository's purpose as a companion to the HashiCorp tutorial.

<https://github.com/hashicorp/learn-terraform-multicloud-kubernetes/tree/main>

Abstract—Infrastructure-as-code (IaC) is the DevOps tactic of managing and provisioning infrastructure through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools. From a maintenance and evolution perspective, the topic has picked the interest of practitioners and academics alike, given the relative scarcity of supporting patterns, best practices, tools, and software engineering techniques. **Using the data coming from 44 semi-structured interviews to senior developers of as many companies, in this paper we shed light on the state of the practice in the adoption of IaC and the key software engineering challenges in the field. Particularly, we investigate (i) how practitioners adopt and develop IaC, (ii) which support is currently available, i.e., the typically used tools and their advantages/disadvantages, and (iii) what are the practitioner's needs when dealing with IaC development, maintenance, and evolution.** Our findings clearly highlight the need for more research in the field: the support provided by currently available tools is still limited, and developers feel the need of novel techniques for testing and maintaining IaC code.

Index Terms—Infrastructure-as-Code; DevOps; Software Maintenance & Evolution; Cloud Automation;

Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry

Michele Guerriero,¹ Martin Garriga,² Damian A. Tamburri,³ Fabio Palomba⁴

¹Politecnico di Milano, Italy

²Jheronimus Academy of Data Science & Tilburg University, The Netherlands

³Jheronimus Academy of Data Science & Eindhoven University of Technology, The Netherlands

⁴University of Zurich, Switzerland

michele.guerriero@polimi.it, m.garriga@uvt.nl, d.a.tamburri@tue.nl, palomba@ifi.uzh.ch

Challenges of IaC

Table IV: Differences between IaC code and standard code (Q5), ordered by the number of mentions (#) among all answers.

Diff	Description	#
Impossible Testing	The lack of standard practices for testing and proper (maybe local) testing environments makes testing painful	8
Declarative	Standard programming is in terms of class, functions, flow. With IaC the reasoning is declarative, i.e., express what is needed, not how to do it.	7
Graph vs. Tree model	Production code is shaped as a graph of modules whereas IaC code resembles a tree of nodes.	7
Impossible Debugging	Similar to testing, the lack of standard practices and the distributed nature makes debugging painful.	6
IaC error-prone	The tools for code checking are more shallow for IaC (e.g., lack of type-checking).	5
Longer feedback loop	The developer has to wait for the whole infrastructure to be deployed in order to understand the correctness of the IaC code.	3
Unmaintainable	As the infrastructure evolves and the computer resources changes, IaC code cannot cope with those on a seamless way.	2

M. Guerriero, M. Garriga, D. A. Tamburri and F. Palomba, "Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 580-589, doi: 10.1109/ICSME.2019.00092.

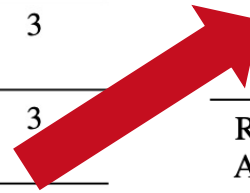
Bad & best practice

Table V: Bad practices when Developing IaC (Q6) ordered by number of mentions.

Bad-practice	Description/Effects	#
Hardcoding	Hardcoding values on the script such as credential or constants.	5
Too Polyglot	Using many languages in interrelated node definitions.	4
Blob blueprints	Generating too large scripts.	3
Non idempotent code	Writing scripts with side effects can lead to undesired states.	3
Poor documenting	Hinders understandability and maintainability	3
Manual infrastructure	Some parts of the configuration are made manually, outside of IaC scripts.	2
Nodes too deep	The tree of nodes generated from a single script is too deep.	2

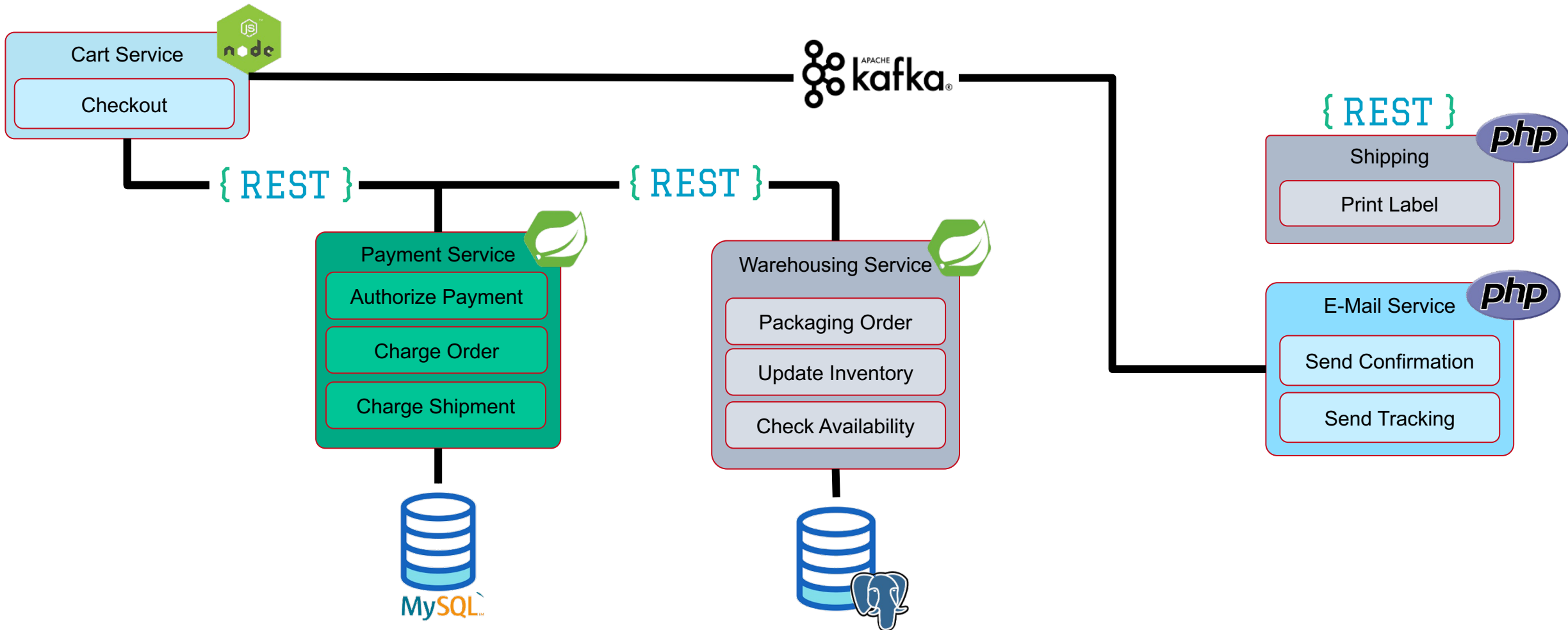
Table VI: Best practices for developing IaC (Q7), ordered by number of mentions.

Best-practice	Description	#
Secret-Injection	Keep all contents of the blueprint or IaC scripts parametric so that the orchestrator or users can inject the desired results at will	12
Break-fast	program the infrastructure to be buildable as fast as possible and hopefully as fast-breaking as possible furthermore, infrastructure circuit-breakers are needed to minimize waste	8
Reuse by Abstraction	Making templates and scripts also recall each-other to allow for interdependency but also interchangeability and possibly reuse, e.g., object orientation	6
Low-Nesting	Keep nodes nesting to at most one level of recall (i.e., tree of height 2)	5

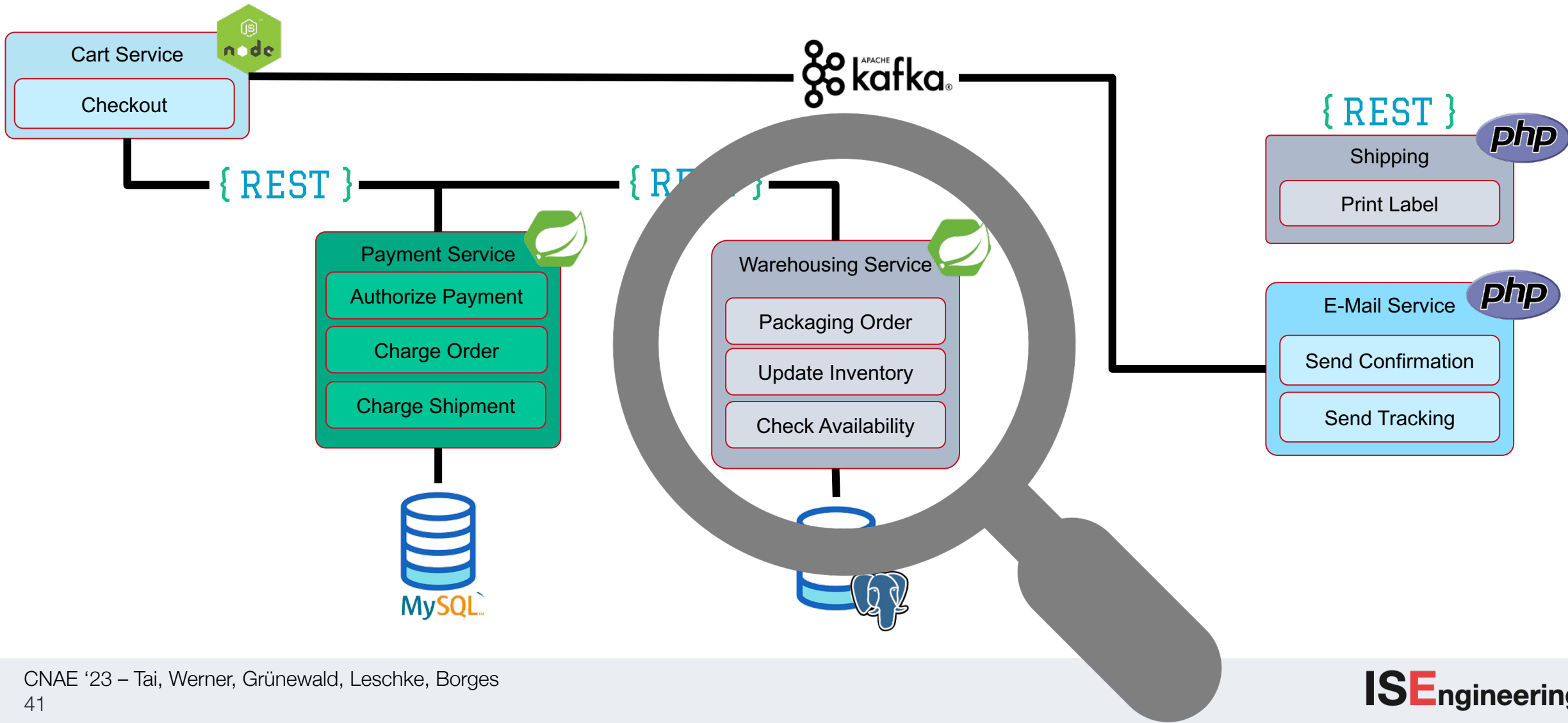


M. Guerriero, M. Garriga, D. A. Tamburri and F. Palomba, "Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry," 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME), Cleveland, OH, USA, 2019, pp. 580-589, doi: 10.1109/ICSME.2019.00092.

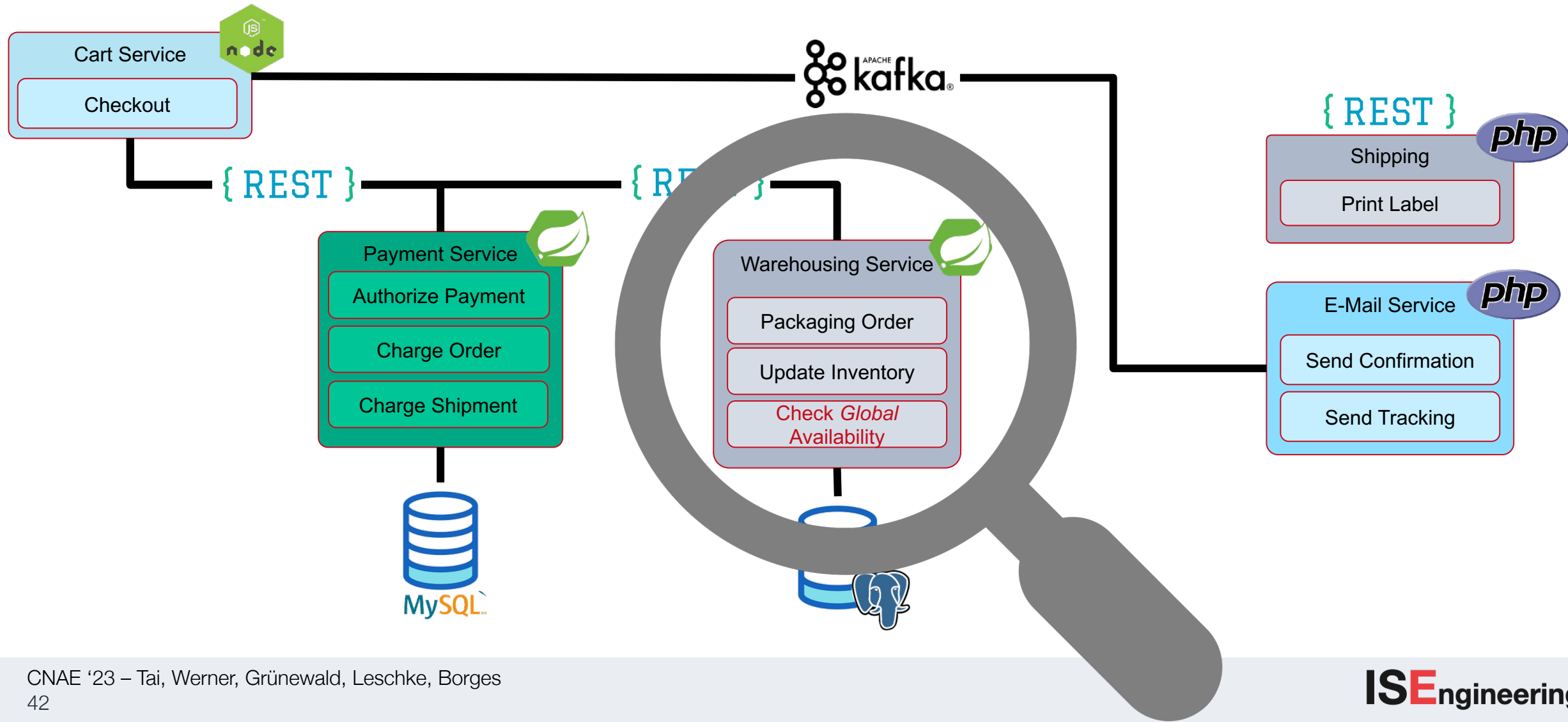
Web shop example

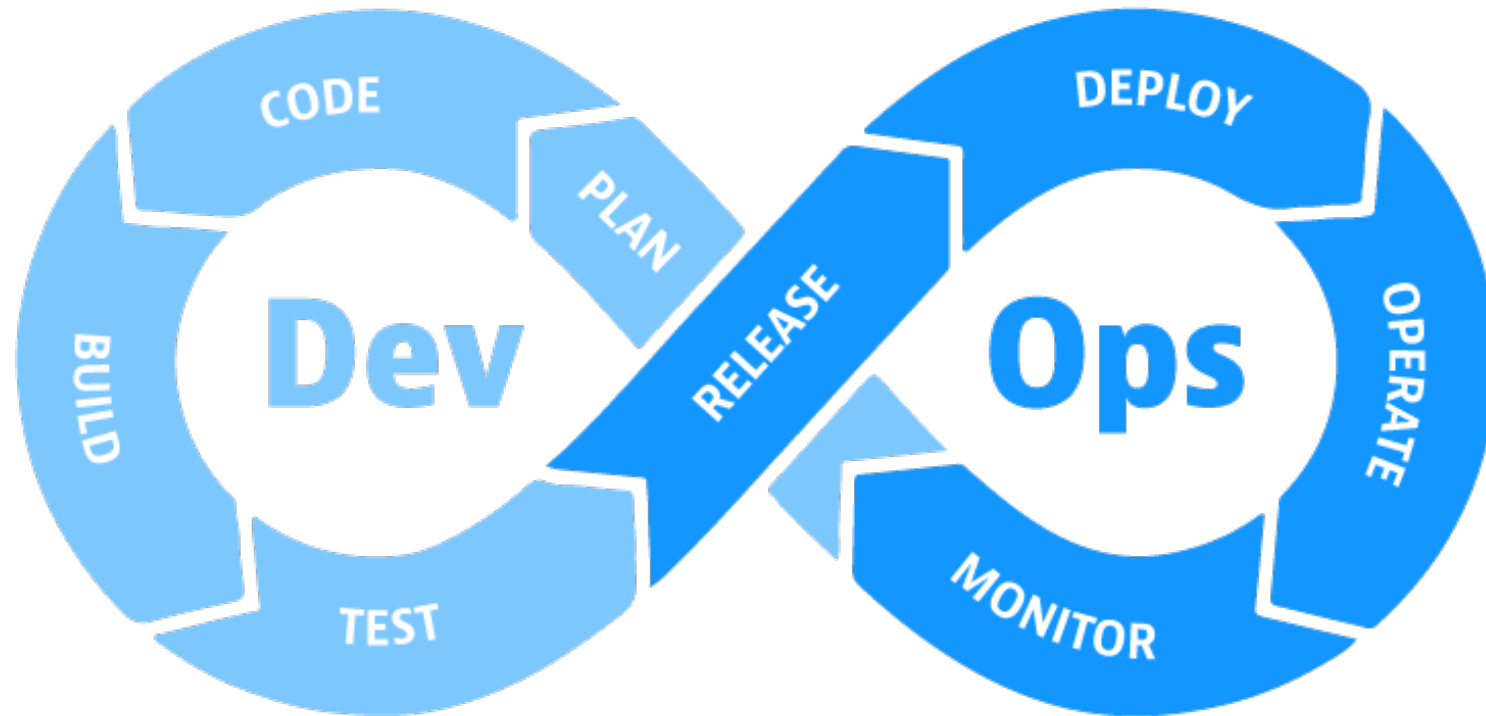


Web shop example



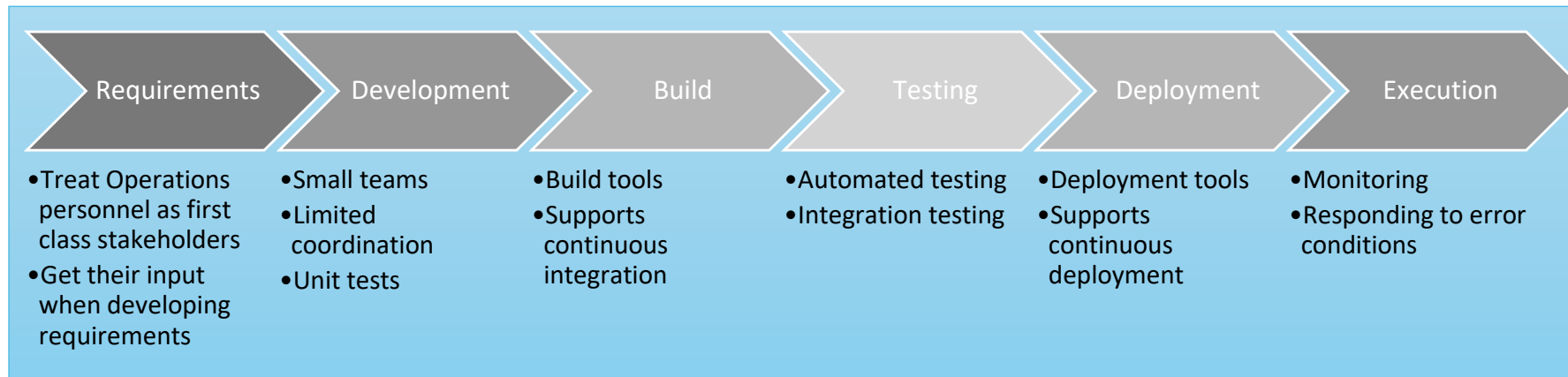
Introducing a new service version





“DevOps is a set of practices intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality.”

DevOps Practices



Treat **Ops as first-class citizens** throughout the lifecycle – e.g., in requirements elicitation

- Many decisions can make operating a system harder or easier
- Logging and monitoring to suit Ops

Make **Dev more responsible for relevant incident handling**

- Shorten the time between finding and repairing errors

DevOps Practices

Make the deployment pipeline **continuous**

- Commits trigger sequence of automatic build, testing, deployment: Continuous Deployment
- This process is used by all teams and for all services/components
- ➔ Automated Deployment Pipeline

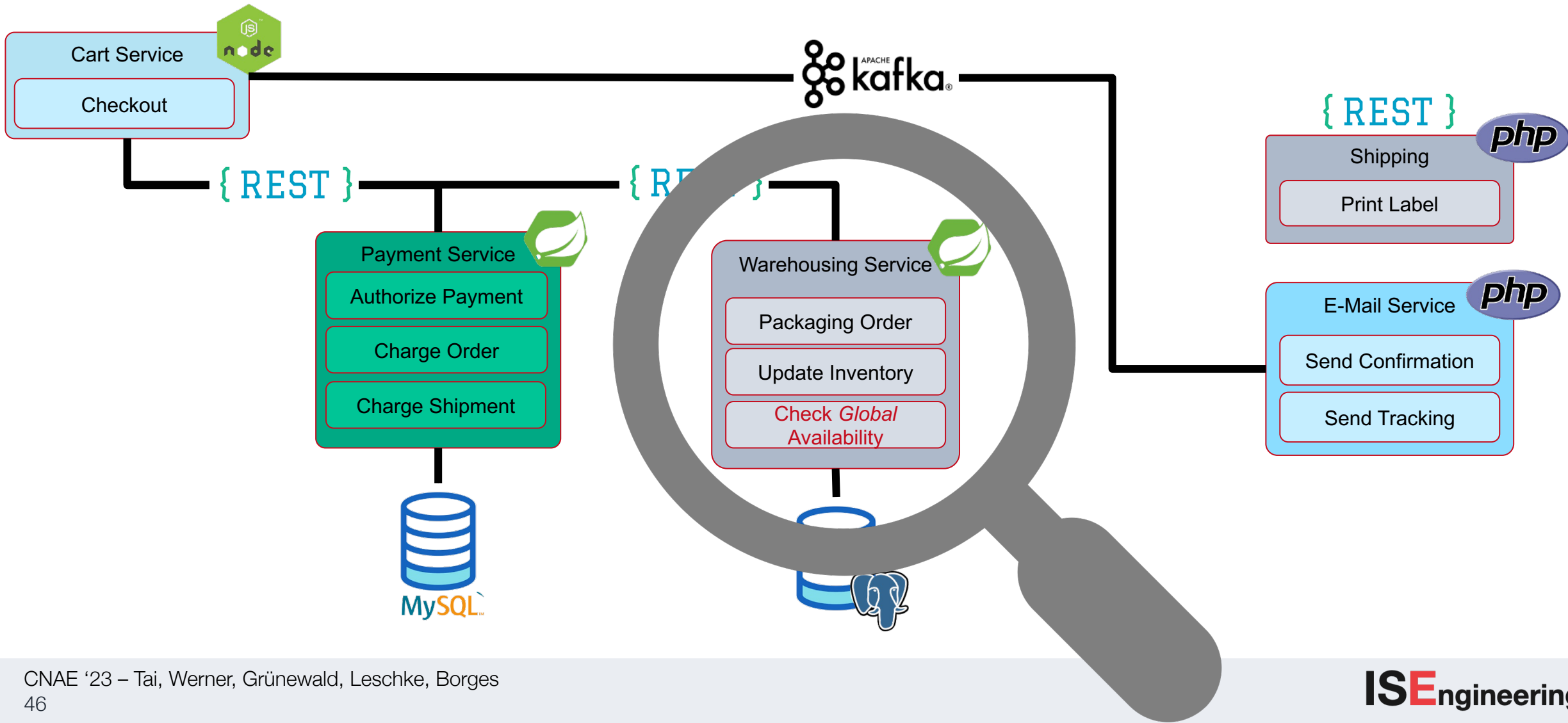
Develop infrastructure code with the same set of practices as application code

- “Infrastructure as Code”: using IaaS APIs etc. to automate creation of environments
- Misconfiguration can derail your application
- ➔ Version Control

Tests are critical to keep quality of an application high

- No “skipping” of tests
- ➔ Automated and incremental testing

Introducing a new service version

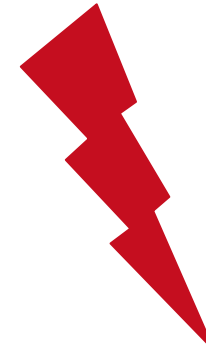


What would you do?



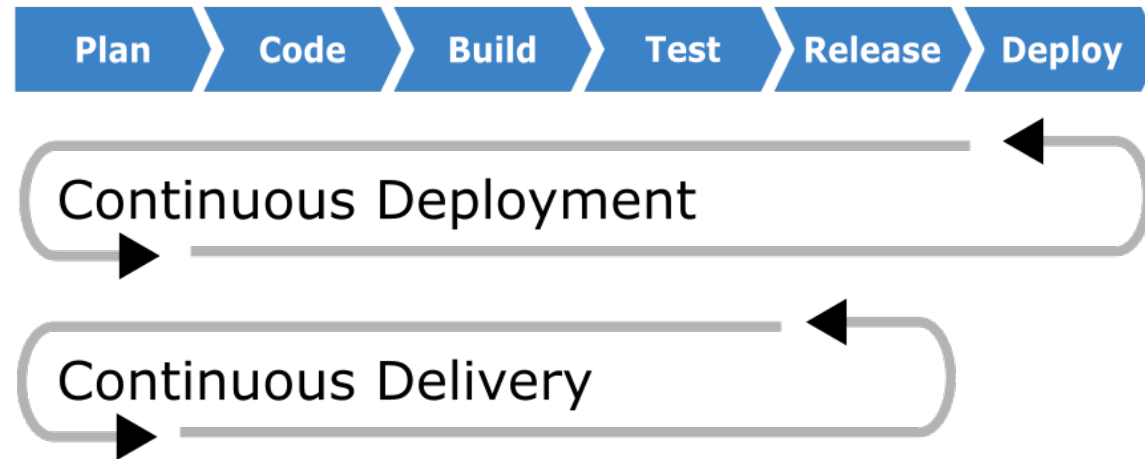
```
for s in www1 www2 db1 db2 cache1:  
do  
done
```

```
ssh -t root@${s} git pull && flask run
```



```
terraform init  
terraform plan  
terraform apply
```


Continuous Delivery & Deployment



Continuous Delivery /
Continuous Deployment = Infrastructure Automation
+ Automated Roll-out

Roll-out Strategies for Continuous Deployment

Continuous Deployment requires **automated backup or roll-back**, if something goes wrong

- The faster a roll-back possible, the better
- Must consider other, currently underway changes to the system, e.g. scaling operations
- **Transactional configuration** of infrastructure

Consider possible **conflicts and compatibility**

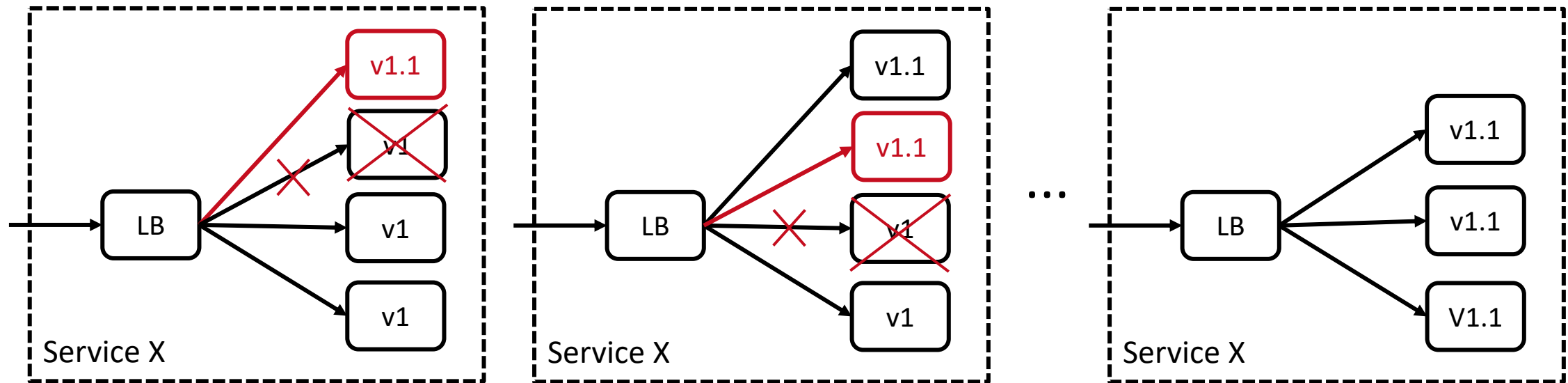
- E.g., can the same client, in parallel, use both v1 and v2? Is the client aware of this? What about consistency in the database if that happens?

Because releases are frequent, a **downtime** during release is **generally not acceptable**

- Consider additionally required resources to avoid bottlenecks
- Roll-out strategies can **incorporate testing approaches**

Gradual/Incremental Roll-out

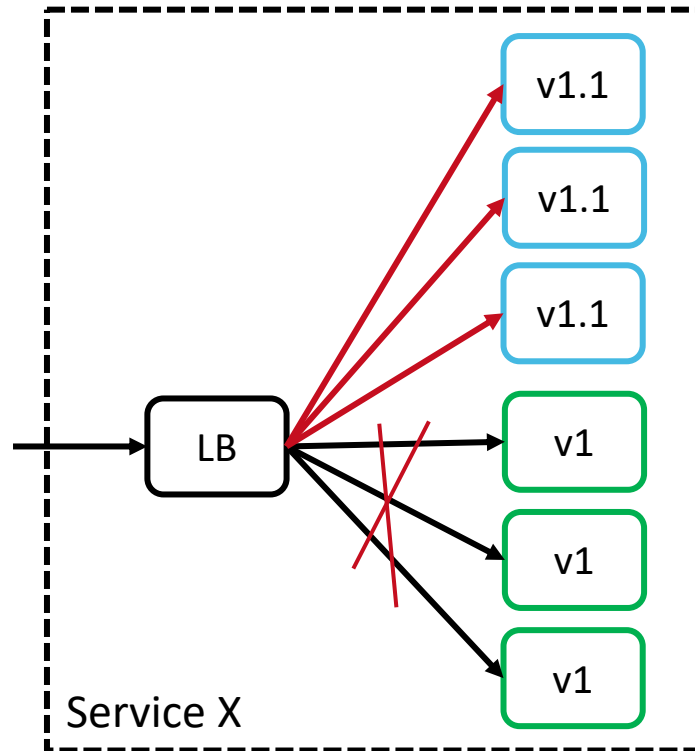
- Gradual roll-out is **resource-efficient** (only +1 instance)
- Requires **compatibility** of dependencies between versions
- Stopping** the roll-out is **easy**
- Going back** to the previous version **can be slow**



— Changes for release

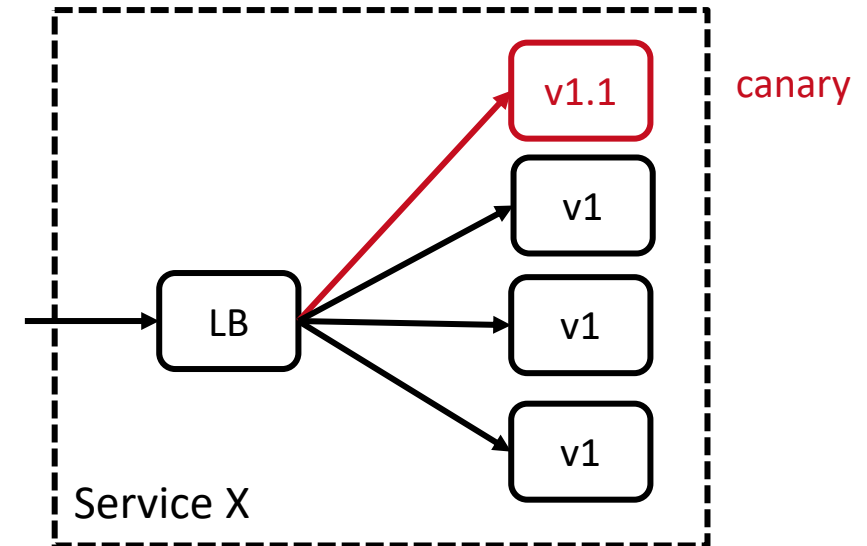
Advanced roll-out strategies

Blue-Green-Deployment (also called Red-Black)



- Switch to blue version, once all instances are up
- Good if dependencies are updated at the same time (blue slice and green slice, no compatibility problems)
- Fast roll-back, but many additional resources

Canary Release



- Observe the canary
- Usually combined with conditional gradual roll-out
- Can deploy one or multiple instances
- Roll-back is trivial

— Changes for release

Challenges of Canary deployments

Canary deployment impacts regular production deployment

- Re-configuration of infrastructure required
- Side-effects
 - Observation of canary
 - Performance issues



Minimize interference

Repeatability of canary assessment

- Request variability
 - Production systems vary in load, state and scale
- Host variability
 - Available resources vs. used resources
 - Side-effects from other components on the same host



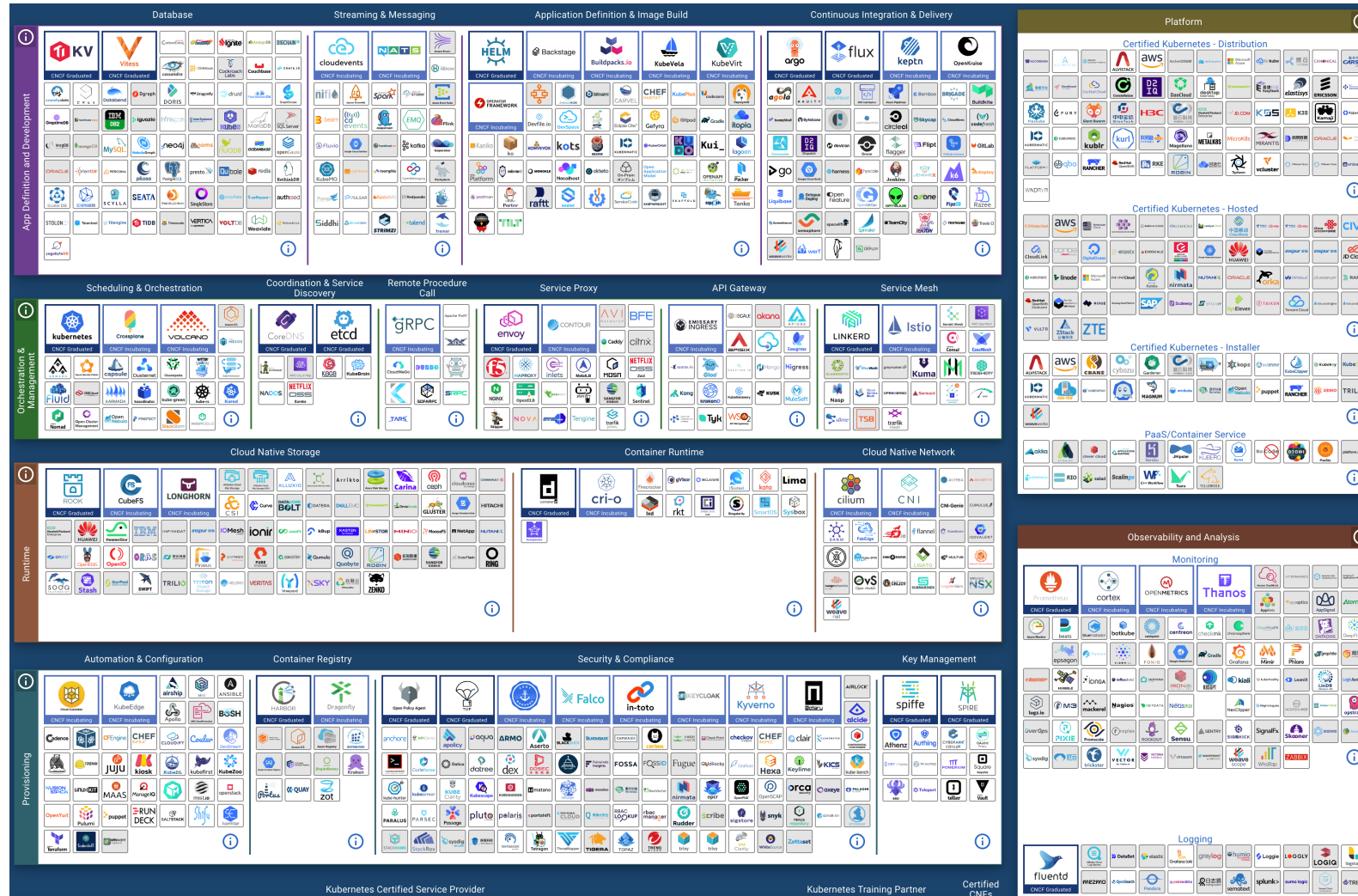
Maximize validity

Comparison of different deployment strategies

Strategy/ Parameters	No Downtime	Real traffic Testing	User targeting	Infra Cost	Rollback duration	Negative User impact	Complexity
Recreate				\$			
Blue/Green				\$\$\$			
Canary				\$			
A/B Testing				\$			
Shadow				\$\$\$			

<https://www.opsmx.com/blog/advanced-deployment-strategies-devops-methodology/>

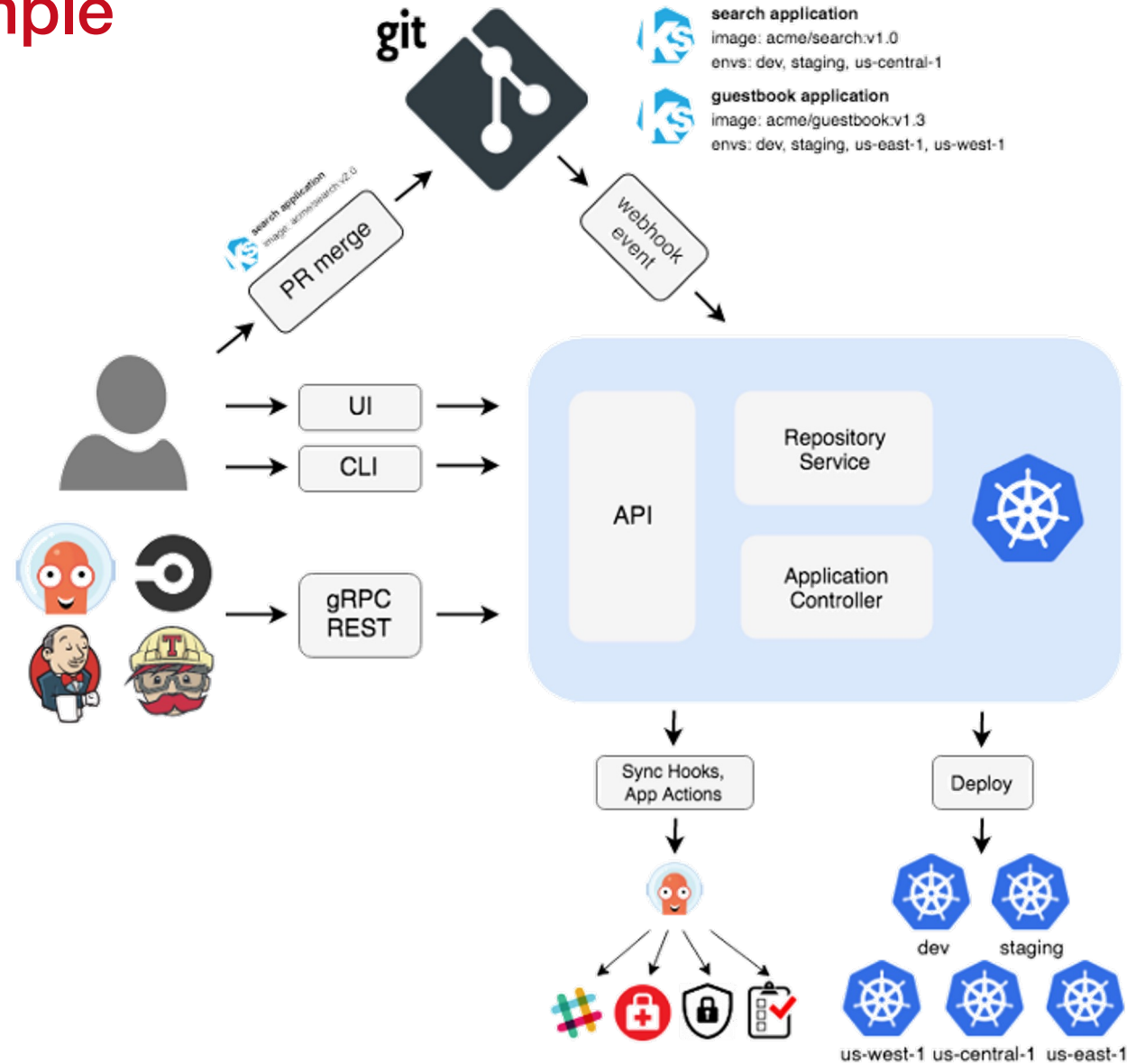
CNCF Landscape



CI/CD

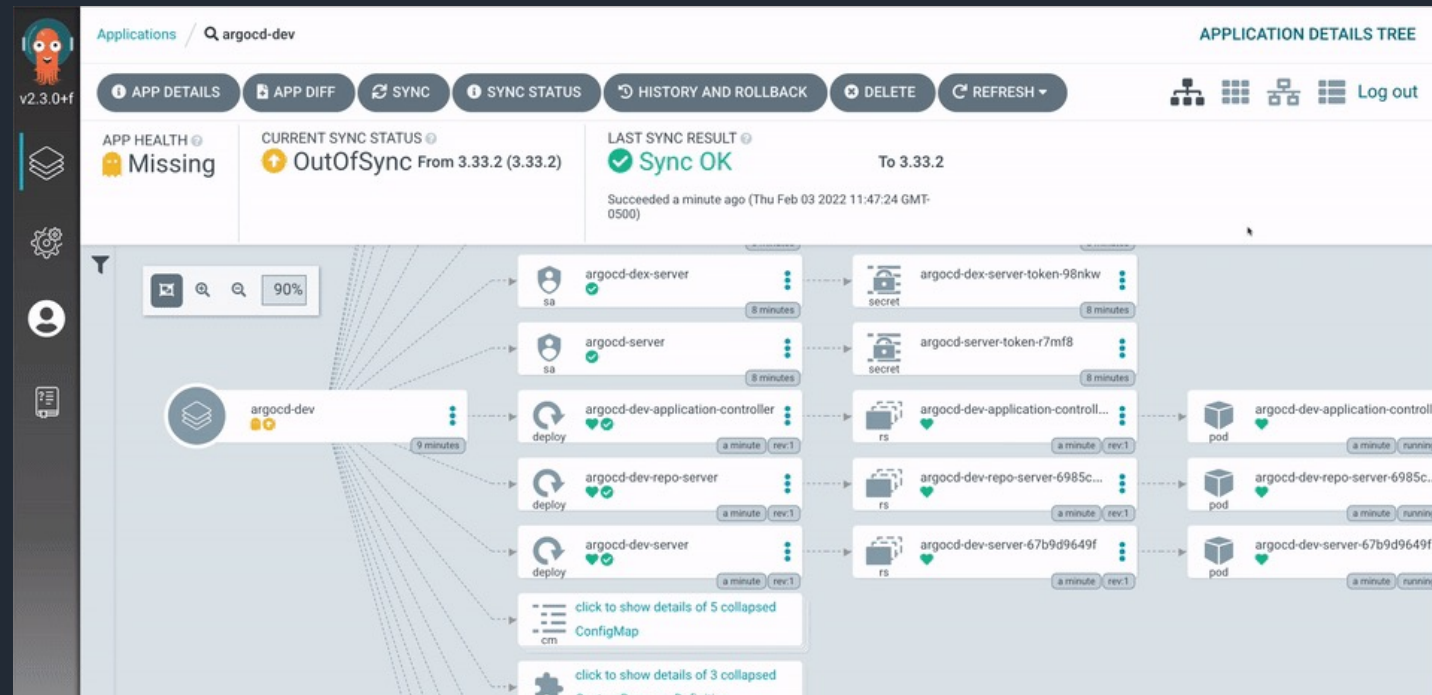


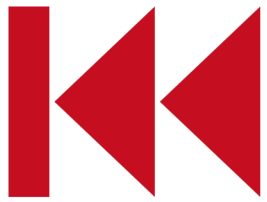
ArgoCD example



Case study: ArgoCD

- Automated deployments to specified target environments (e.g., test, staging, production)
- PreSync, Sync, PostSync hooks to support complex application rollouts
- SSO, Access tokens, Audit trails...
- Metrics, Webhooks
- Git synchronization (GitOps)

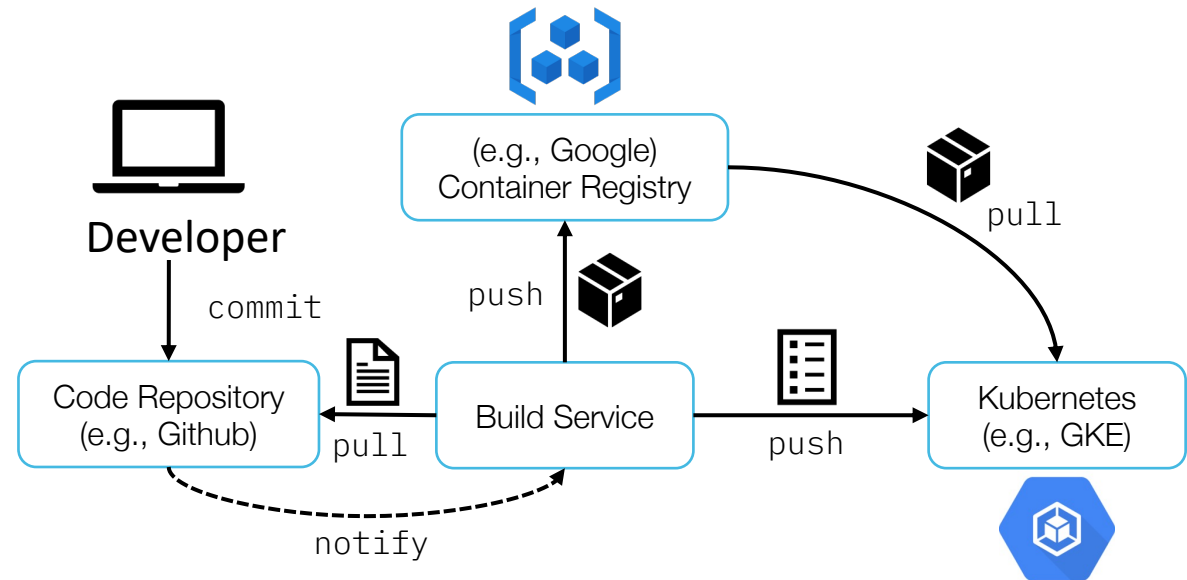


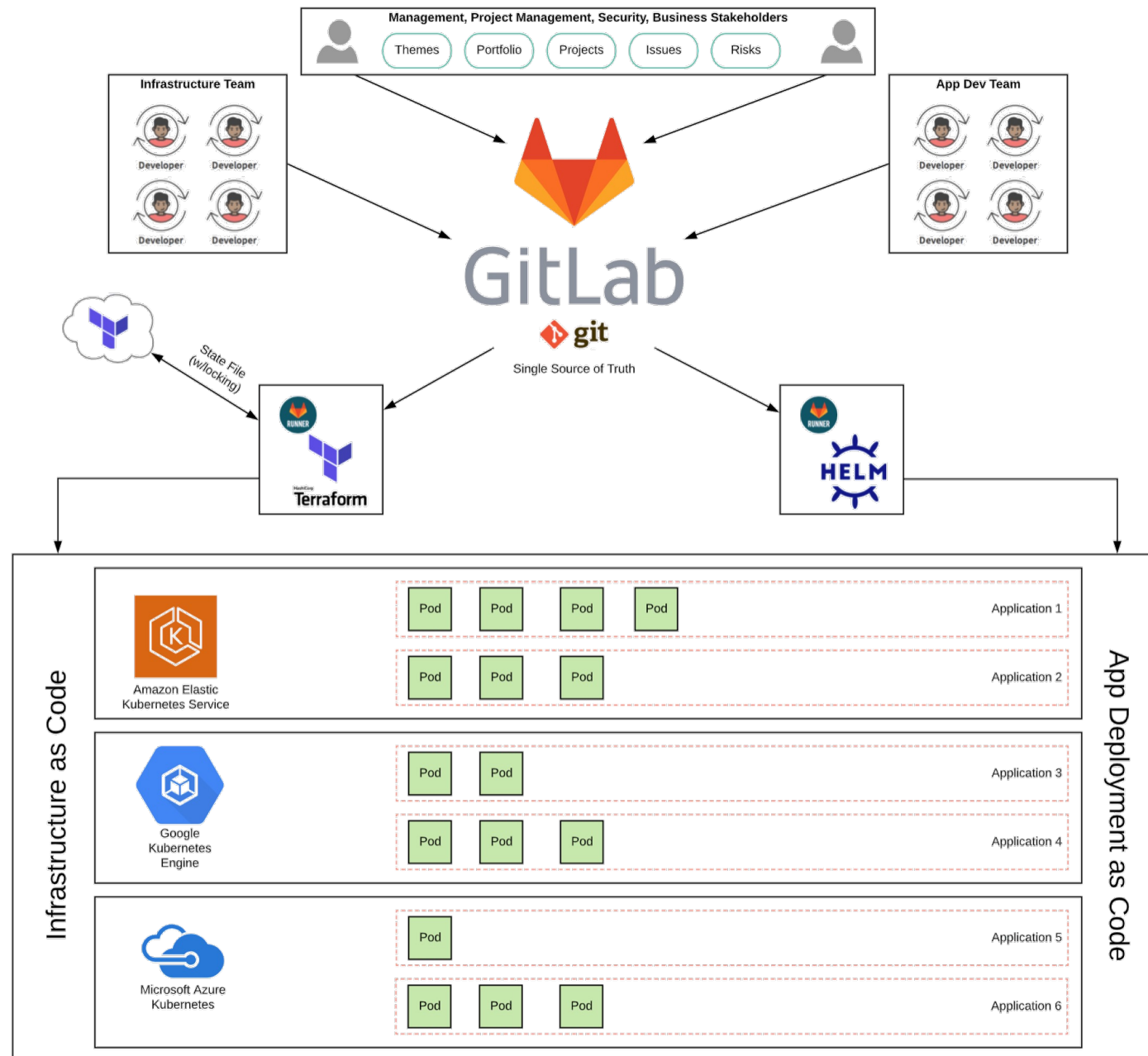


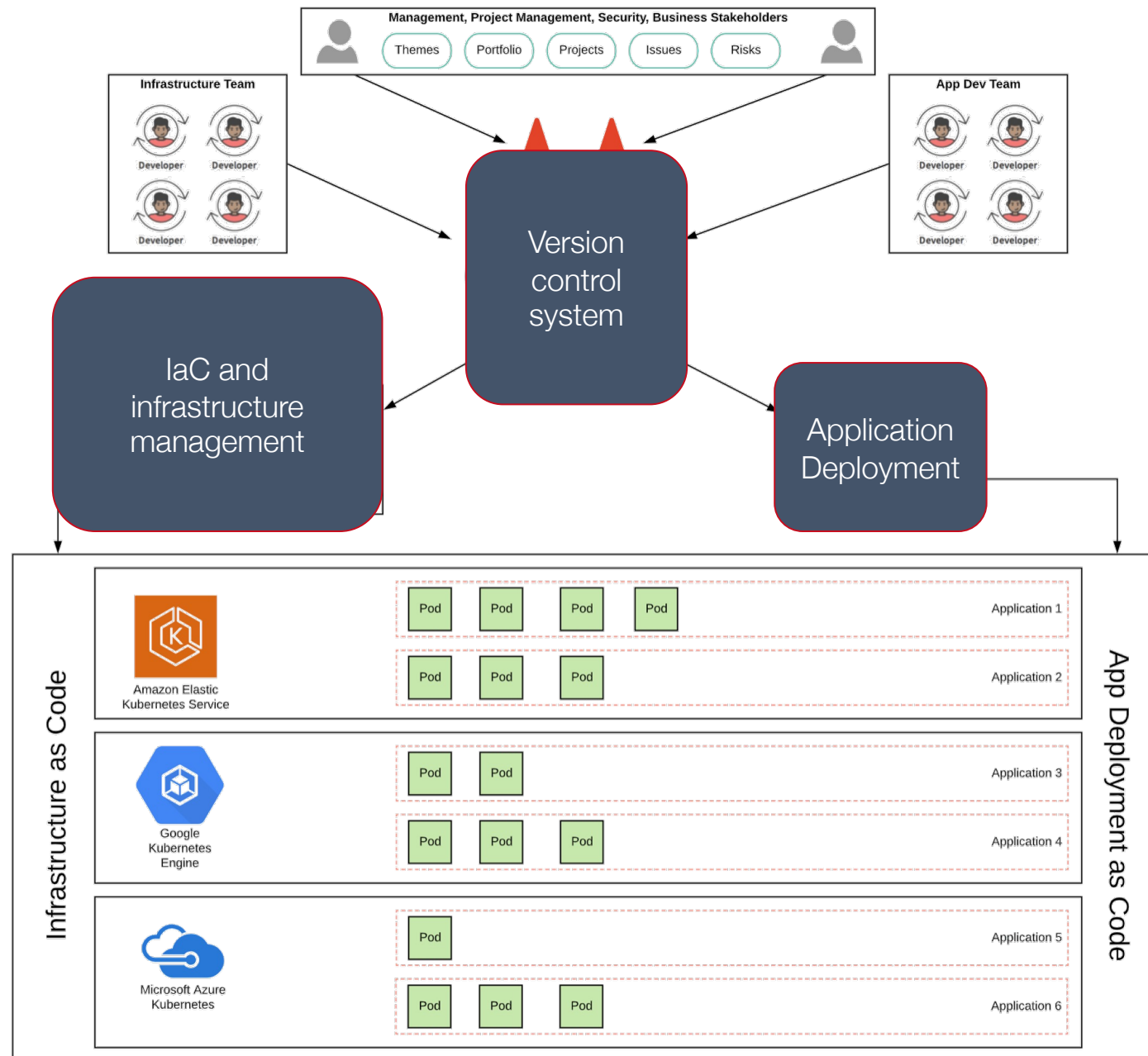
Continuous Development

- Different degrees of automation and customization
- Insight into ongoing release processes
- Automated roll-back in case of failures
- Different roll-out strategies

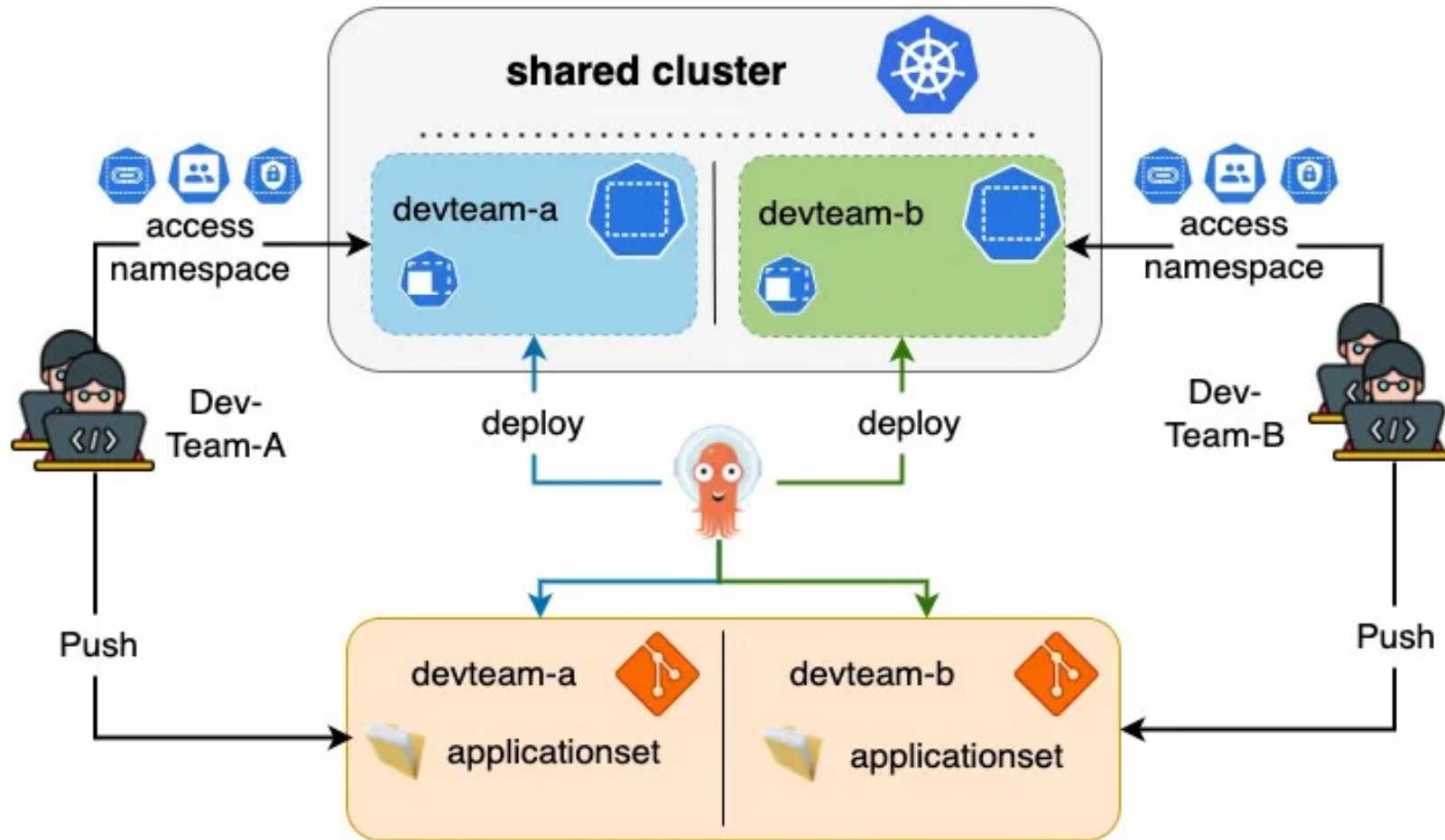
Works well for simple rollouts and "smaller" architectures!



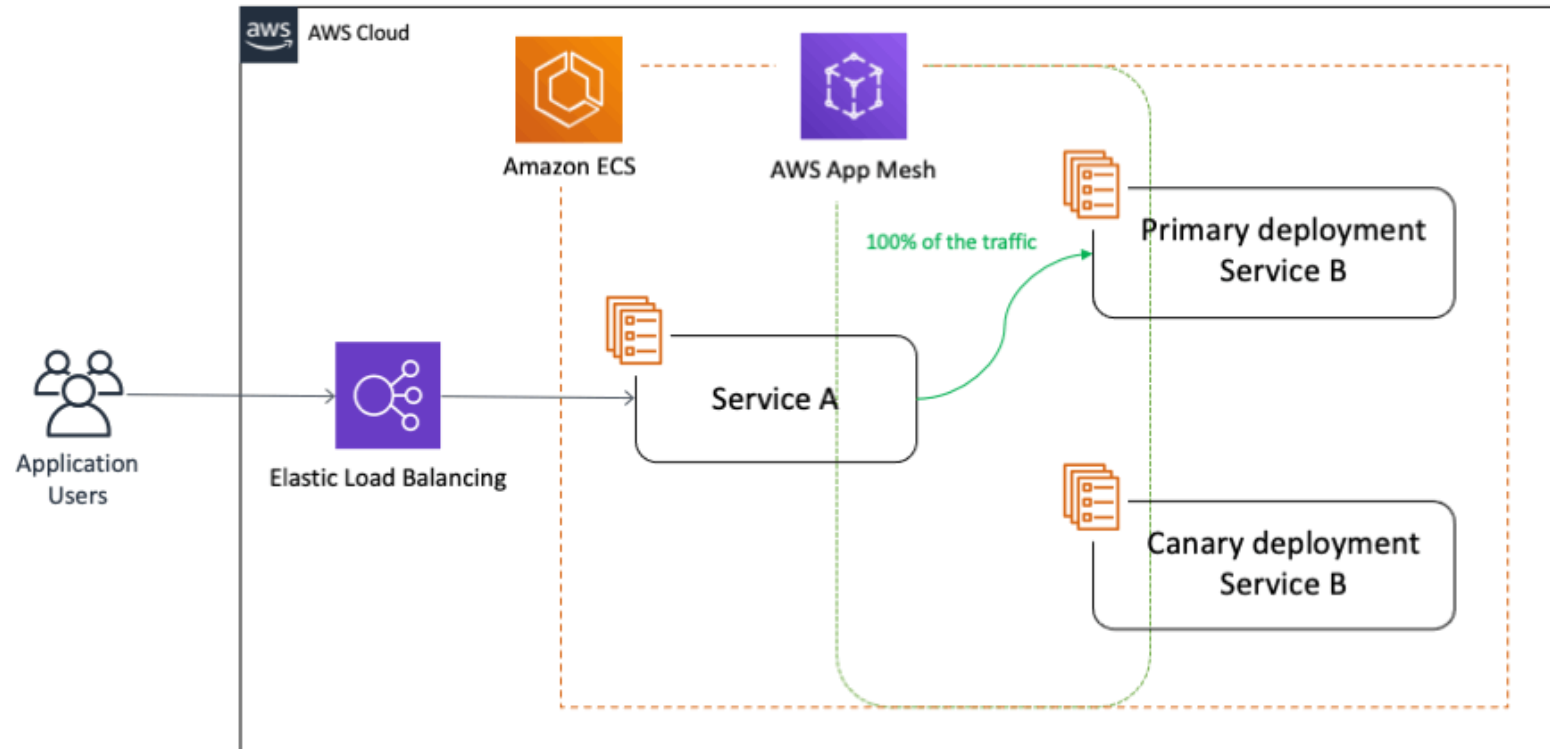




Multi-tenancy GitOps

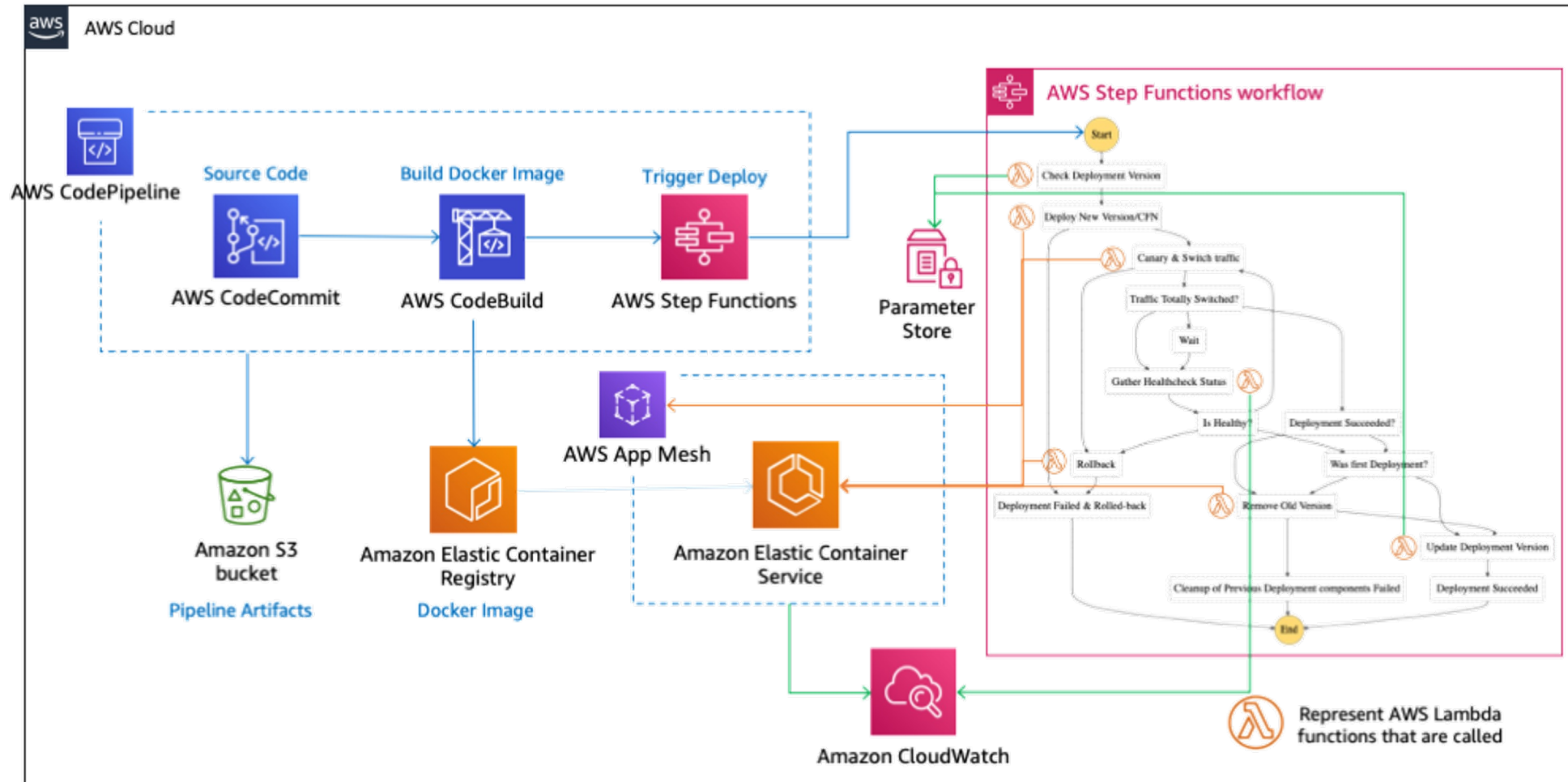


Case study: AWS AppMesh



<https://aws.amazon.com/blogs/containers/create-a-pipeline-with-canary-deployments-for-amazon-ecs-using-aws-app-mesh/>

Case study: AWS AppMesh



<https://aws.amazon.com/blogs/containers/create-a-pipeline-with-canary-deployments-for-amazon-ecs-using-aws-app-mesh/>

Case study: AWS AppMesh

In the background, a AWS Step Functions state machine orchestrates multiple Lambda functions for the canary deployment

23.05.
Serverless I

30.05.
Serverless II

Export Layout

Details Step input Step output

Name	Type
Update Deployment Version	Task

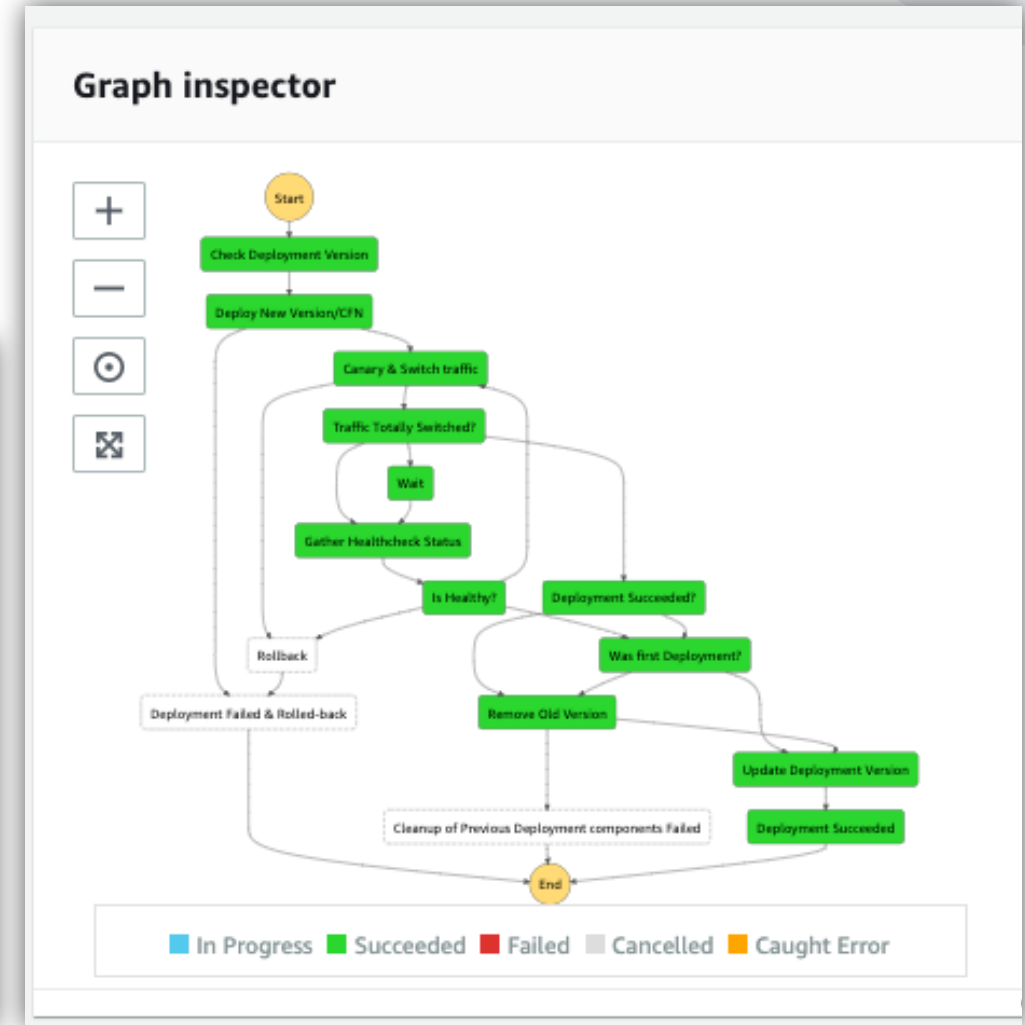
Status

✔ Succeeded

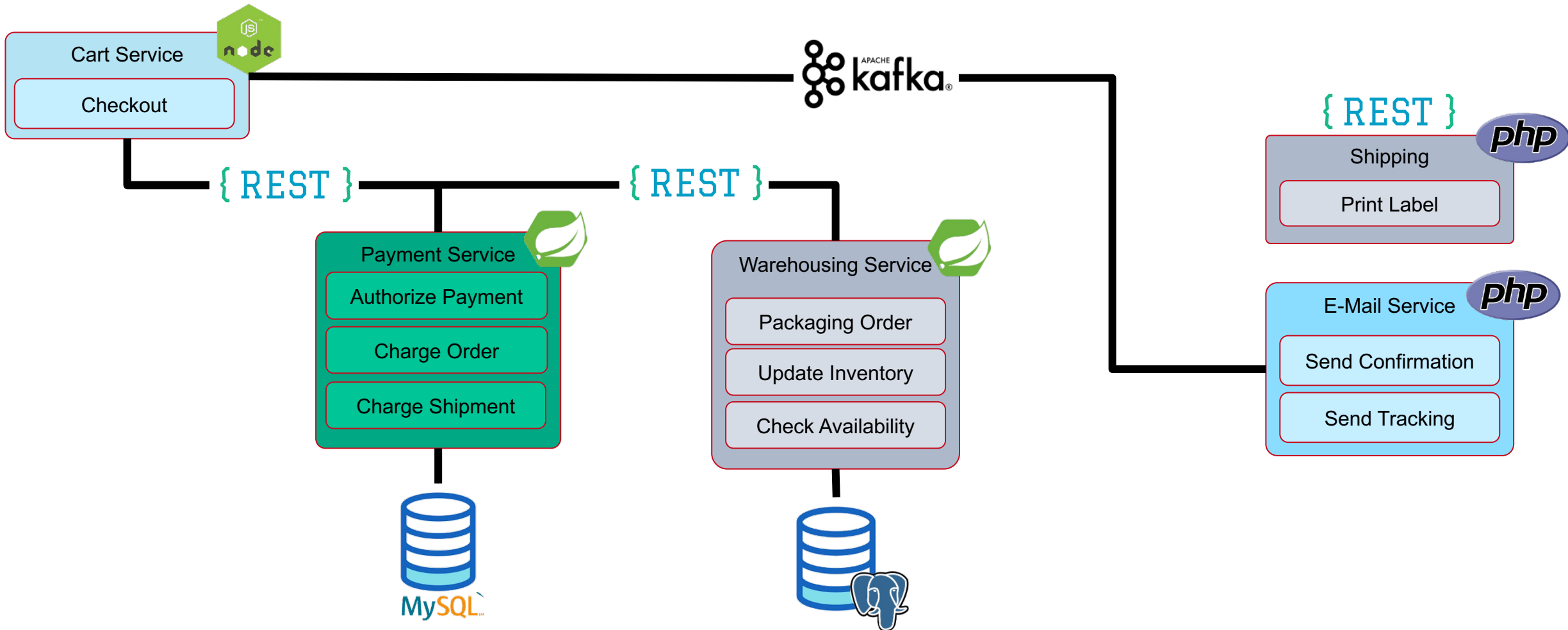
Resource

arn:aws:lambda:us-east-1:██████████:function:██████████

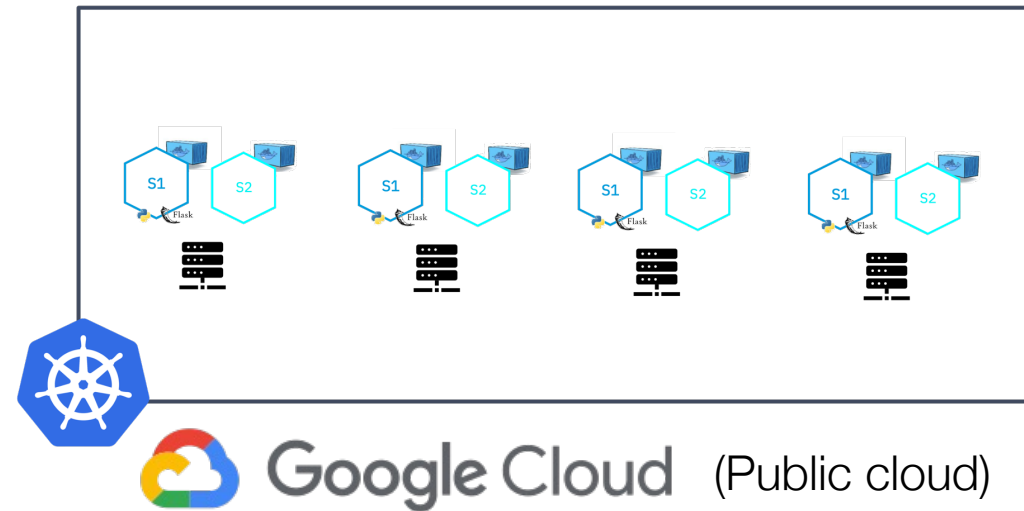
CloudWatch Logs



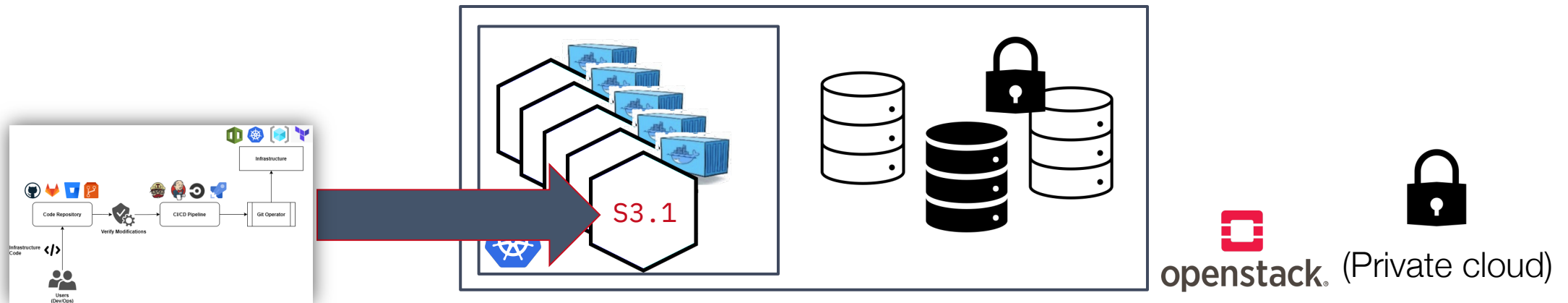
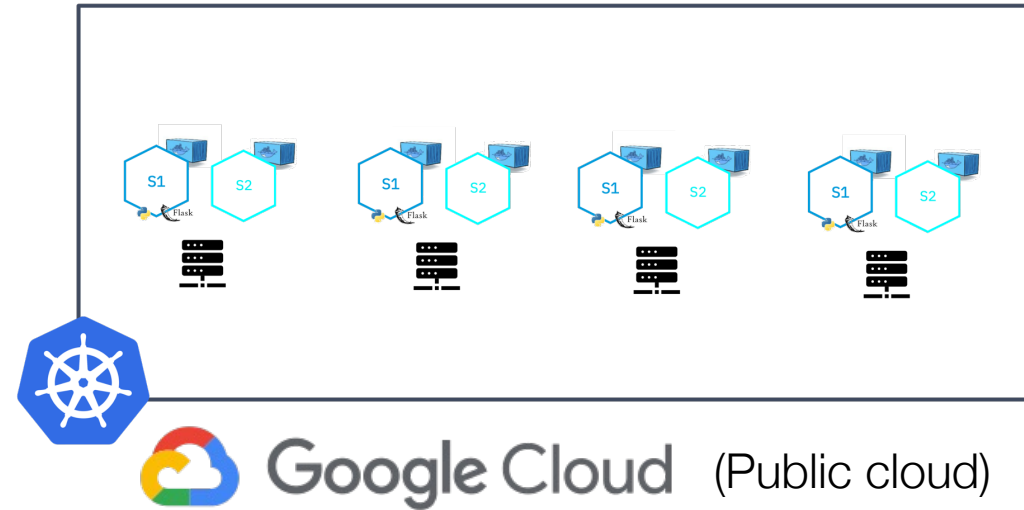
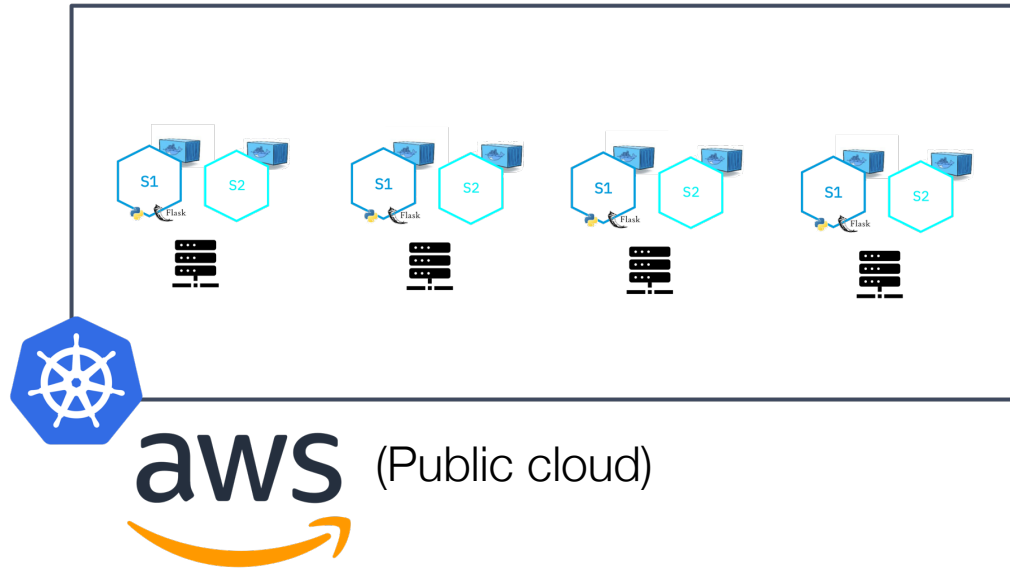
Web shop example



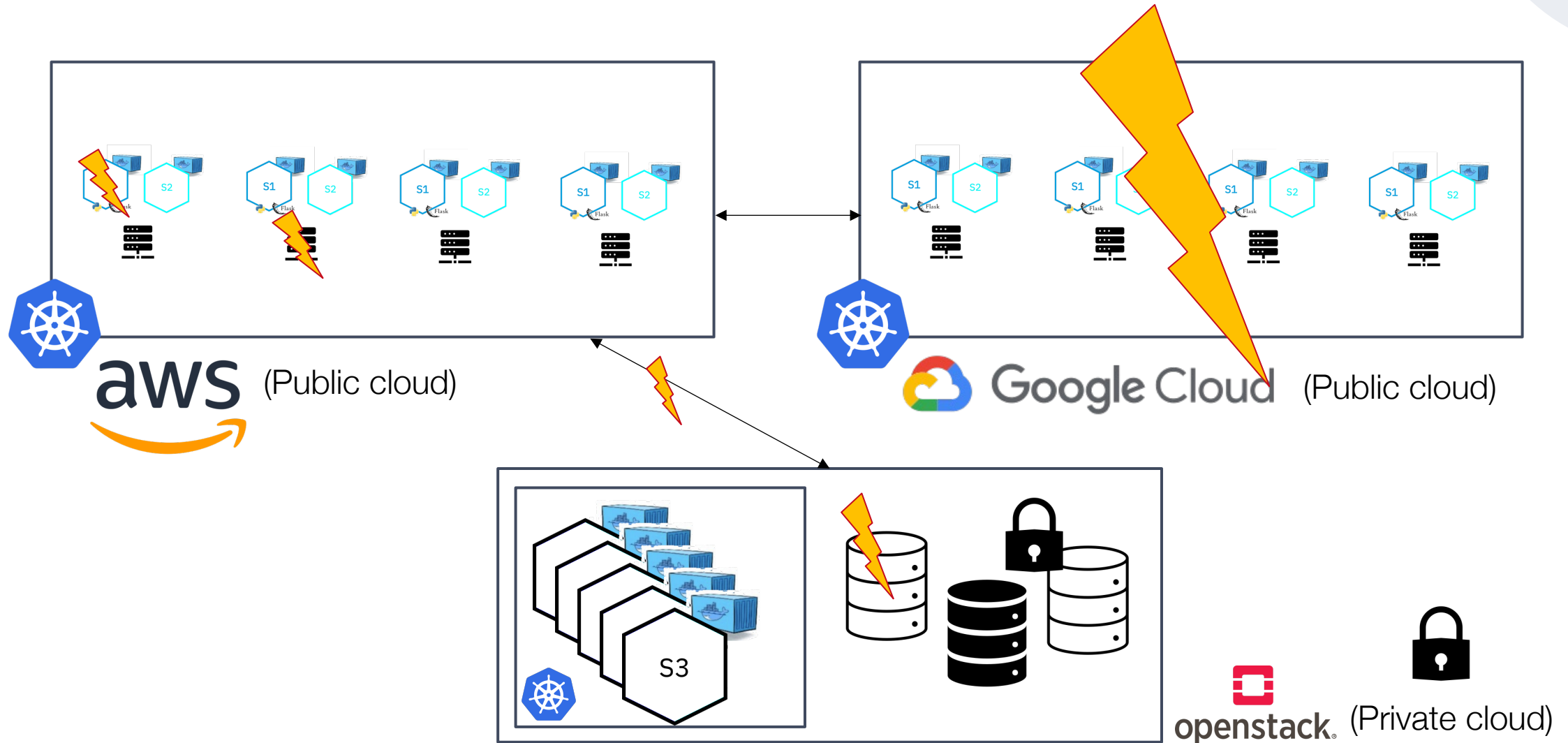
Hybrid & Multi-cloud infra management essentials



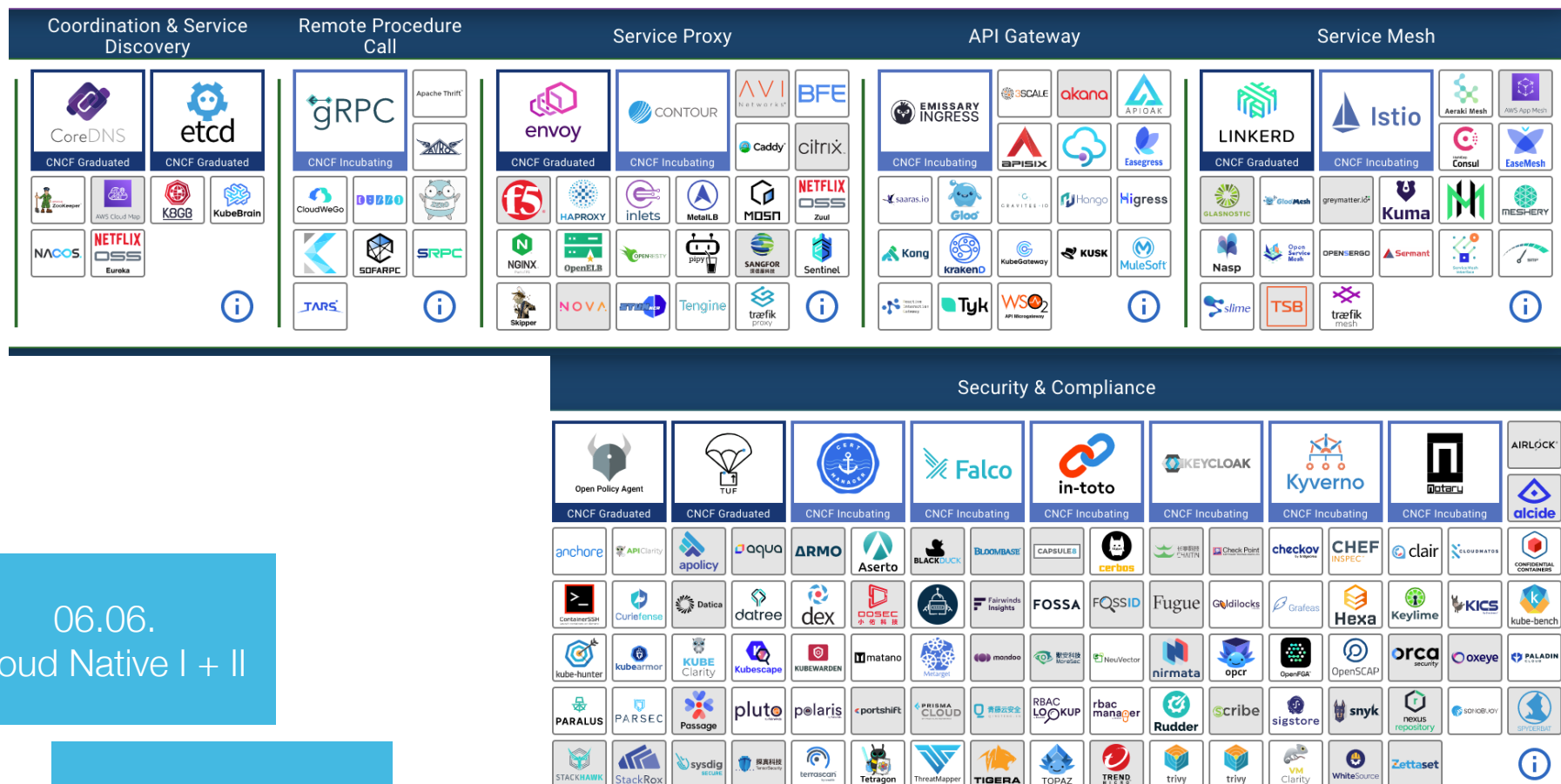
Release strategies



Remember: Errors and failures are the norm



Many remaining challenges



06.06.
Cloud Native I + II



Guest lectures

Recommended tooling

Ansible

<https://www.ansible.com/>

Terraform

<https://www.terraform.io/>

ArgoCD

<https://argo-cd.readthedocs.io/>

Flux/Flagger (Progressive Delivery)

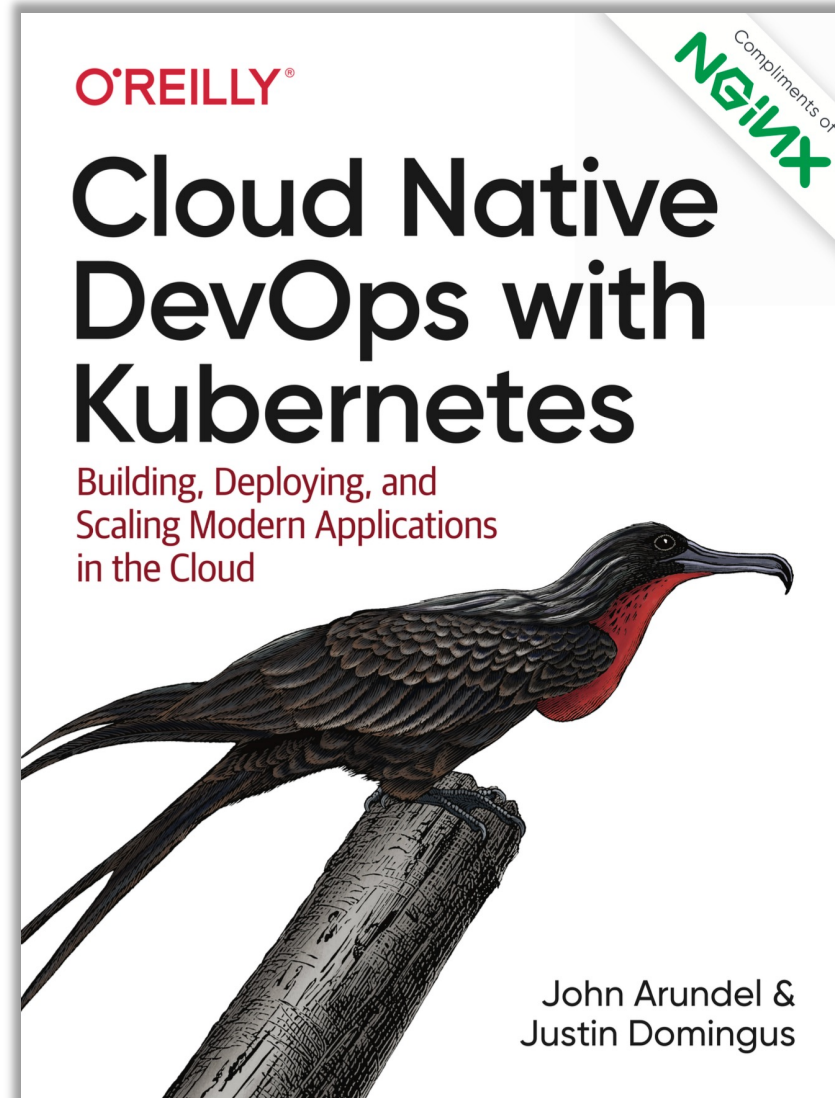
<https://fluxcd.io/flagger/>

AWS Cloud Formation

<https://aws.amazon.com/cloudformation/>

Many more

<https://collabnix.github.io/kubetools/>



This element signifies a tip or suggestion.

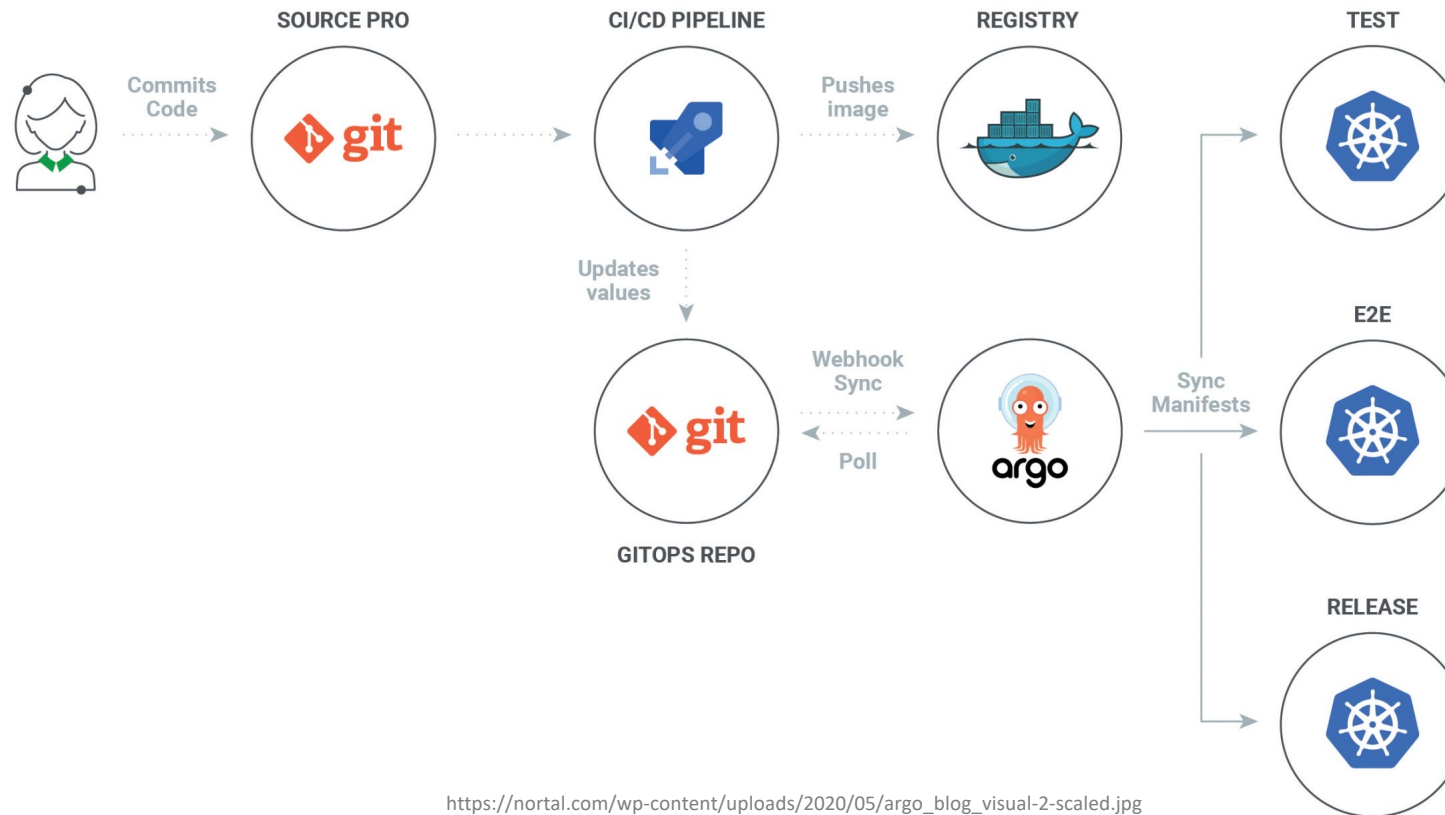


This element signifies a general note.



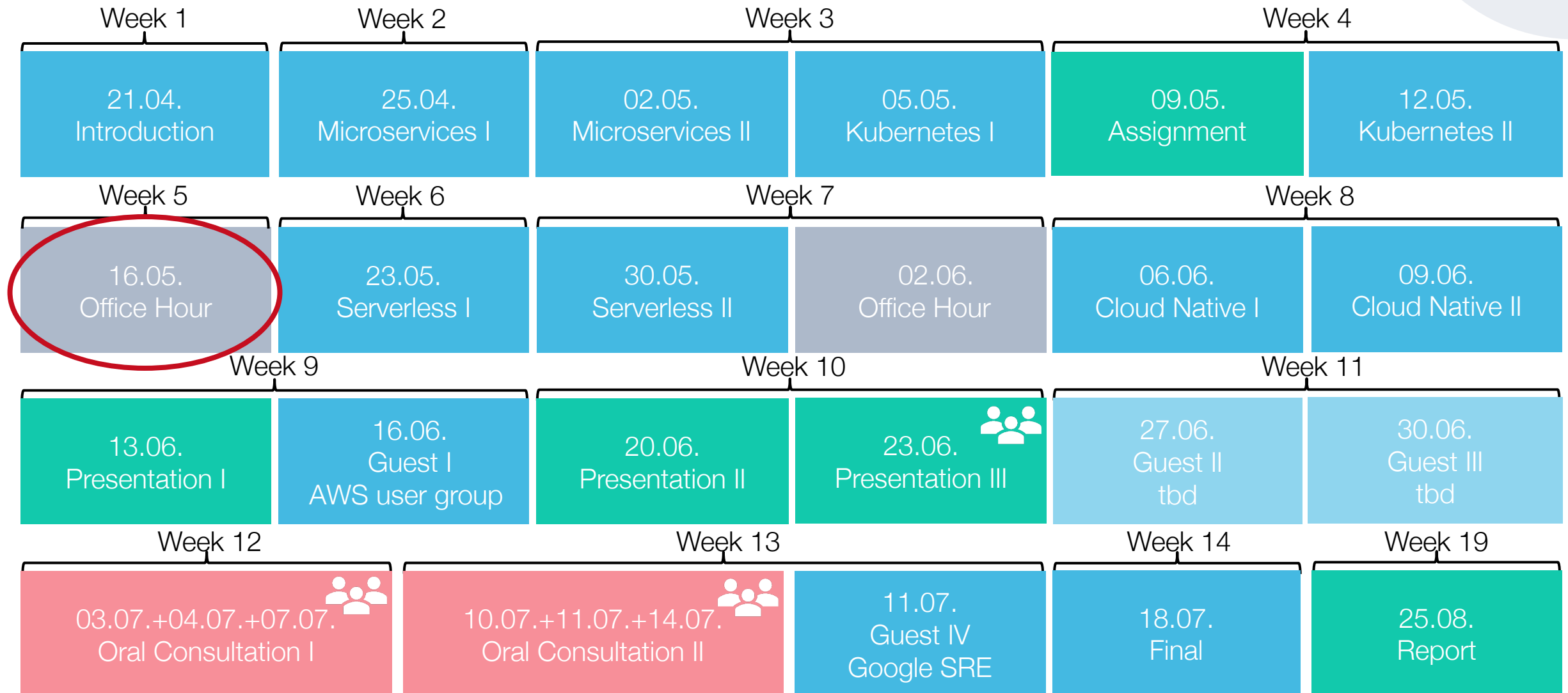
This element indicates a warning or caution.

Job Interview Question

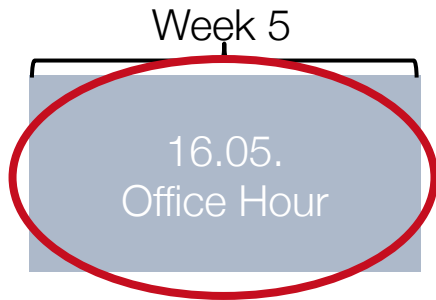


Your external consultant proposes this workflow to you, how would you respond?

Schedule (still preliminary)



Schedule



For all virtual meetings, we will use the following Zoom-Link:

<https://tu-berlin.zoom.us/j/65284334867?pwd=QXBNV0xNb3ZNVmZ3SIRGZ21Uekg0QT09>

Meeting-ID: 652 8433 4867

Kenncode: 375255

Container Orchestration and Management II

CNAE – Cloud Native Architecture & Engineering



Thank you!



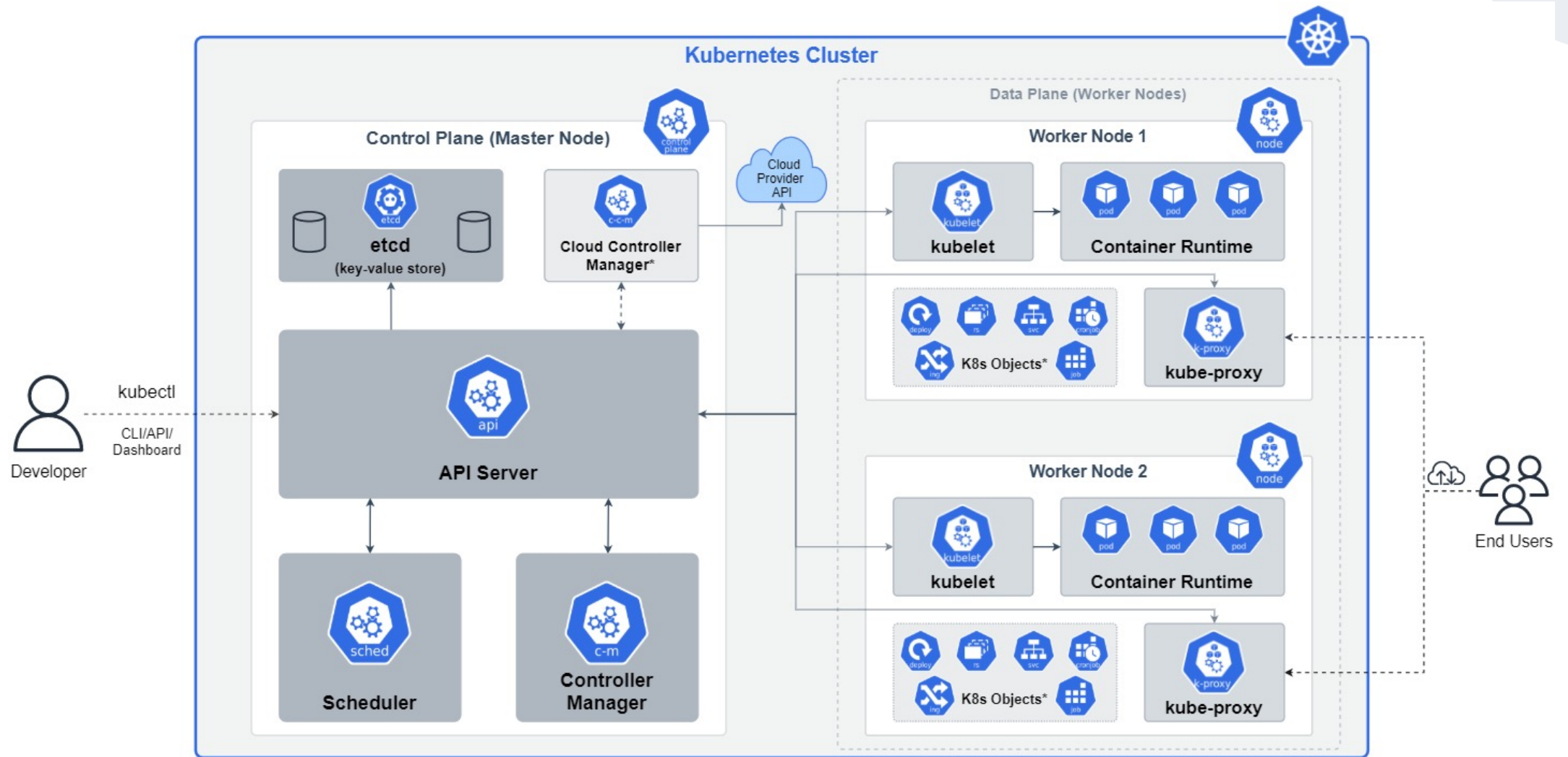
Prof. Dr.-Ing. Stefan Tai
Sebastian Werner, Elias Grünewald, Nicola Leschke, Maria Borges

Information Systems Engineering
TU Berlin

Appendix



Kubernetes - Overview



<https://medium.com/devops-mojo/kubernetes-architecture-overview-introduction-to-k8s-architecture-and-understanding-k8s-cluster-components-90e11eb34ccd>